(3) 1148
1 2

Honeywell

HONEYWELL
INFORMATION
SYSTEMS INC.

TITLE: No. 43A177741

ENGINEERING PERFORMANCE SPECIFICATION (EPS-1)
645 Follow-On CPU

Total Pages 196

REVISION RECORD

REVISION LETTER	DATE	PAGES AFFECTED	APPROVALS	AUTHORITY
A	Issued			

PRELIMINARY
NOT FOR
FABRICATION

HONEYWELL PROPRIETARY

The information contained in this document is proprietary to Honeywell Information Systems Inc. and is intended for internal Honeywell use only. Such information may be distributed to others only by written permission of an authorized Honeywell official. This restriction does not apply to vendor proprietary parts that may be disclosed in this

TABLE OF CONTENTS

1. GENERAL DESCRIPTION---------- 3

1.1 INTRODUCTION---------- 3
1.2 SCOPE---------- 3
1.3 APPLICABLE DOCUMENTS---------- 3
1.4 SPECIAL NOTATION---------- 4

2. FUNCTIONAL CAPABILITIES---------- 5

2.1 PROCESSOR ORGANIZATIONS---------- 5

2.1.1 Control Unit---------- 6
2.1.2 Operations Unit---------- 6
2.1.3 Decimal Unit---------- 7
2.1.4 Appending Unit---------- 7
2.1.5 Physical Description---------- 7

2.2 645F OPERATIONAL MODES---------- 7

2.2.1 GECOS Operational Modes---------- 7
2.2.2 MULTICS Operational Modes---------- 8
2.2.3 Privileged Instructions---------- 8
2.2.4 Address Preparation---------- 9
2.2.5 Fault Selection---------- 10
2.2.6 Interrupt and Fault Handling---------- 10
2.2.7 Processor Mode Determination and Alteration---------- 10

2.3 ADDRESS PREPARATION AND MODIFICATION---------- 14

2.3.1 Terms and Definitions---------- 14
2.3.2 Segmentation Concepts---------- 17
2.3.3 Segmentation Implementation in the 645F---------- 19

2.3.3.1 Current Procedure Segment Accessing---------- 20
2.3.3.2 Direct Segment Accessing---------- 20
2.3.3.3 Indirect Segment Accessing---------- 24

2.3.4 Paging Concepts---------- 24
2.3.5 Paging Implementation in the 645F---------- 28

2.3.5.1 Paged Object Segments---------- 28
2.3.5.2 Paged Descriptor Segments---------- 31

2.3.6 Segmentation and Paging Hardware Requirements---------- 31

2.3.6.1 Variances from the 645-Processor---------- 31
2.3.6.2 SDW Format---------- 38
2.3.6.3 PTW Format---------- 41
2.3.6.4 The Descriptor Segment Base Register (DSBR)---------- 42
2.3.6.5 The Procedure Pointer Register (PPR)---------- 43
2.3.6.6 The Temporary Pointer Register (TPR)---------- 44
2.3.6.7 The Eight Pointer Registers (PRn)---------- 45
2.3.6.8 The SDW Associative Memory Assemblies (SDWAM)---------- 47
2.3.6.9 The PTW Associative Memory Assemblies (PTWAM)---------- 49

2.3.7 Absolute Address Preparation Mode------- 51
2.3.8 Append Address Preparation Mode--------- 52
2.3.9 The Address Preparation Sequence-------- 52

2.3.9.1 Computed Address Generation---- 52
2.3.9.2 Append Address Preparation Sequences---------- 57

2.4 APPENDING UNIT ACCESS CONTROL---------- 69

2.4.1 Segment Boundary Control---------- 69
2.4.2 The MULTICS Ring Structure---------- 70

2.4.2.1 Theory and Definition---------- 70
2.4.2.2 Ring Structure Hardware Implementation---------- 73

2.5 INSTRUCTION MODIFICATION OF APPENDING UNIT REGISTERS IN MULTICS OPERATIONAL MODE---------- 81

2.5.1 Data Movement Instructions---------- 81
2.5.2 CALL Instruction---------- 81
2.5.3 RTCD Instruction---------- 81
2.5.4 RCU Instruction---------- 83
2.5.5 TSS Instruction---------- 84
2.5.6 All Other Transfer Instructions--------- 84
2.5.7 The ITS Modifier---------- 85
2.5.8 ITP Modifier---------- 86

3. CUSTOMER INTERFACE---------- 87

3.1 PROGRAM COMPATIBILITY---------- 87
3.2 DATA FORMATS---------- 87
3.3 INSTRUCTION WORD FORMAT---------- 92
3.4 INSTRUCTION DESCRIPTIONS---------- 93

3.4.1 Data Movement Instructions---------- 93
3.4.2 Transfer of Control Instructions------- 124
3.4.3 Special Instructions---------- 128
3.4.4 Modified H6000 Instructions---------- 134

3.5 INDICATOR REGISTER---------- 157
3.6 645F PROCESSOR FAULTS AND INTERRUPTS---------- 157

3.6.1 GECOS Mode---------- 157
3.6.2 MULTICS Mode---------- 157

3.7 645F MULTICS PROCESSOR FAULTS---------- 157

3.7.1 Fault Categories---------- 157
3.7.2 Fault Descriptions---------- 159
3.7.3 Fault Priority---------- 165
3.7.4 Fault Recognition---------- 165
3.7.5 Fault Vector Address---------- 166
3.7.6 Fault Sequence---------- 166
3.7.7 Status Safestore and Restore---------- 168

3.7.7.1 Group I Faults---------- 168
3.7.7.2 Group II Faults---------- 168
3.7.7.3 Group III Faults---------- 168
3.7.7.4 Group IV Faults---------- 168
3.7.7.5 Group V Faults---------- 169
3.7.7.6 Group VI Faults---------- 169
3.7.7.7 Group VII Faults---------- 170

3.7.8 645F Processor Interrupts---------- 170

3.7.8.1 External Interrupts---------- 170
3.7.8.2 External Faults---------- 171
3.7.8.3 Interrupt Inhibit Bit---------- 171
3.7.8.4 Lock-Up Fault---------- 171
3.7.8.5 DIS---------- 172
3.7.8.6 Interrupt Sampling---------- 172
3.7.8.7 Interrupt Vector Address------ 173

3.8 CONFIGURATION CONTROL---------- 173

3.8.1 Address Assignment Switches---------- 173
3.8.2 Interlace Switches---------- 174
3.8.3 Port Enable Switches---------- 174
3.8.5 Size Switches---------- 174
3.8.6 Processor Number Switches---------- 175
3.8.7 Fault Base Switches---------- 175
3.8.8 Initialize and Clear Pushbutton--------- 175

3.9 OPERATORS PANEL---------- 175

3.9.1 Ready/Trouble Indicator---------- 176
3.9.2 Audible Alarm---------- 176

3.10 MAINTAINABILITY AIDS---------- 176

3.10.1 Maintenance Panel---------- 176
3.10.2 History Register---------- 177
3.10.3 Program Accessible T&D Aids---------- 177

3.11 MEASUREMENT AIDS---------- 177

4. HARDWARE INTERFACES---------- 177

4.1 SYSTEM PORT MODULARITY AND INTERFACES---------- 177

4.1.1 Summary of 645F System Controller Interface Lines---------- 178

4.1.1.1 Bidirectional Signals---------- 178
4.1.1.2 Signals to System Controller--- 178
4.1.1.3 Signals to Processor---------- 178

4.2 OPERATIONS WITH THE 645F SYSTEM CONTROLLER----- 178

4.2.1 Addressing---------- 178
4.2.2 Key/Lock Logic---------- 179

4.3 INTERFACE ERROR CHECKING---------- 179
4.4 PROCESSOR INITIALIZATION---------- 179
4.5 635 MASTER CONSOLE INTERFACE---------- 179

5. GENERAL DESIGN REQUIREMENTS---------- 180

6. RELIABILITY AND MAINTAINABILITY---------- 180

6.1 RELIABILITY OBJECTIVES---------- 180

6.1.1 Availability---------- 180
6.1.2 Mean Time Between Failures---------- 180

6.2 TRANSIENT ERROR RATE---------- 180
6.3 MEASUREMENT OF TRANSIENT ERROR RATE---------- 180
6.4 MAINTAINABILITY REQUIREMENTS---------- 181

1. GENERAL DESCRIPTION

1.1 INTRODUCTION

The 645F Processor incorporates similar features of the 645 Processor to function as a multiple-access time-shared computer. However, additional hardware features shall be incorporated where significant cost/performance enhancements to MULTICS can be achieved and the impact on software redevelopment would be minimal.

The 645F Processor shall have a Manual Mode switch to allow it to selectively function as a 645F Processor with Extended Instruction Set capability (MULTICS Mode) or an H6000 Series Processor (GECOS Mode).

The 645F Processor will be implemented in the standard H6000 integrated circuit and packaging technology, utilizing MSI technology, where applicable.

1.2 SCOPE

This Engineering Product Specification (EPS) defines and describes the requirements and design specifications of the 645F Processor. This specification describes those functions that differ from the standard H6000 Processor except where some H6000 details are needed to place the 645F in proper perspective.

1.3 APPLICABLE DOCUMENTS

The following documents of the issue in effect on the issue date of this specification shall form a part of this specification to the extent specified herein.

Specifications

43A232717	EPS-1 - 655/EIS Processor
43A219600	EPS-1 - GE-655 System
43A219601	EPS-1 - GE-655 Processor
43A219617	EPS-1 - GE-655 Maintainability
43A177851	EPS-1 - GE-655 General Design Requirements
M50EB00152	EPS - GE-645 Model B Processor
M50EB00621	EPS - GE-615/625/635/635B Processors

Manuals

G0098 GE-645 Processor Reference Manual
CPB-1004 GE-625/635 Programming Reference Manual

Standards

Information Systems Group Standard No. 1.2.01
Information Processing Terminology

IFIP - ICC Vocabulary of Information Processing, North Holland Publishing Company, Amsterdam, 1966.

MIL STD 781A, Reliability Test, Exponential Distribution.

Other Publications

MHDM-14 MULTICS 645 Follow-On Design Memo 14
MHDM-15 MULTICS 645 Follow-On Design Memo 15
MHDM-16 MULTICS 645 Follow-On Design Memo 16
MHDM-17 MULTICS 645 Follow-On Design Memo 17
MHDM-18 MULTICS 645 Follow-On Design Memo 18
MHDM-19 MULTICS 645 Follow-On Design Memo 19
MHDM-20 MULTICS 645 Follow-On Design Memo 20
MHDM-21 MULTICS 645 Follow-On Design Memo 21
MHDM-22 MULTICS 645 Follow-On Design Memo 22

1.4 SPECIAL NOTATION

Throughout this EPS the following symbolic notation is used:

1. A.B is used to denote the quantity B contained in A. For example, PTW.ADDR denotes the ADDR field within the PTW.

2. The double vertical bar (||) is used to denote concatenation (for example, PTW.ADDR||000000).

3. The single vertical bar (|) is used to denote the logical inclusive "OR."

4. C(A) $\Rightarrow$ C(B) is used to denote that the contents of A replace the contents of B.

5. C(A)/C(B) is used to denote that the contents of A temporarily replace the contents of B.

6. $C(A)_{00-05} \xrightarrow{b/cb} C(B)_{00-05}$ is used to denote that the binary bit number designator in A is converted to a character designator which replaces B_{00-01} and a bit designator which replaces B_{02-05}.

7. $C(A)_{00-05} \xrightarrow{cb/b} C(B)_{00-05}$ is used to denote that the character designator in A_{00-01} and the bit designator in A_{02-05} are converted to a bit designator which replaces B_{00-05}.

2. FUNCTIONAL CAPABILITIES

2.1 PROCESSOR ORGANIZATION

The processor shall be organized into four functional units: (1) the Control Unit; (2) the Operations Unit; (3) the Decimal Unit; and (4) the Appending Unit. In addition, the processor shall have a Mode Control Manual switch; in one position it will function as a 645F Processor and in the other position it will function as a H6000 Series Processor. This switch causes the appending unit to be logically added to or removed from the address/data lines, the command lines, and port selection logic to the system controllers. It shall also change the legality of several op codes, faults,etc.

The first three of the aforementioned units shall perform all the functions they perform in the H6000 Processor. In addition, they must perform functions necessary to operate as a 645F Processor and provide interfaces and control signals to the appending unit.

The major functions of each of the 645F Processor Units and their differences from the standard H6000 Processors are briefly summarized as follows:

2.1.1 Control Unit

The control unit of the 645F Processor shall function as a standard H6000 Control Unit with the following modifications:

- Modes of Operation -- As a 645 Processor, the 645F Processor shall use the normal 645 absolute/append word addressing modes and the privileged/unprivileged (645 Master/Slave) instruction execution modes. As an H6000 Processor, the 645F Processor shall be limited to the combinations of H6000 Master (Absolute and Privileged) Mode and H6000 Slave (Append and Unprivileged) Mode.

- Operation Decoding -- In either of the two processor modes (645F or H6000), all instruction operation codes (10 bits) not defined as legal in that mode shall cause an Illegal Procedure Fault if execution is attempted.

- External Fault and Interrupt Recognition -- As a 645 Processor, the control unit shall only recognize the use of bit 28 to inhibit external faults and interrupt recognition as specified in Sections 2.2.6.

- Fault and Fault Control Logic -- As a 645 Processor, the control unit shall recognize up to 27 faults. As an H6000 Processor, the 645F Processor shall only recognize the first 16 faults. (See Section 2.2.5.)

- Storage Protection -- As an H6000 Processor, the 645F Processor shall preserve user integrity via a Base Address Register (BAR). As a 645 Processor, this register will be disabled unless the 645F Processor is internally put in the MULTICS BAR Mode (see Section 2.2).

2.1.2 Operations Unit

The operations unit of the 645F Processor shall function as the H6000 Operations Unit defined by 43A219601, EPS-1, GE-655 Processor.

2.1.3 Decimal Unit

The decimal unit of the 645F Processor shall function identical to the H6000 Decimal Unit as defined by 43A232719, EPS-1, 655/EIS Processor.

2.1.4 Appending Unit

The appending unit performs the system requirements for segmentation and paging of memory. This appending concept is described in Section 2.3. The registers and functions provided by the appending unit are as follows:

- Provide 24 bit addressing
- Contain up to 16 segment descriptor words and 16 page table words on a most recently used basis
- Provides a Descriptor Segment Base Register
- Provides eight Segment Pointer Registers
- Provides ring protection hardware

2.1.5 Physical Description

The 645F Processor physical description will be similar to the H6000 Processor, the only exception being a second configuration panel described in Section 3.8.

2.2 645F OPERATIONAL MODES

This section covers the operational modes of the 645F Processor. The mode of operation the processor may assume is defined by a switch located on the Processor Configuration Panel.

2.2.1 GECOS Operational Modes

The GECOS Operational Modes shall be in effect when the Mode Control Switch is set to the GECOS position. When in this mode, the 645F Processor shall function as a H6000 Series Processor, and all addresses shall be generated in the Absolute Address Preparation Mode as described in Section 2.3.7, special instructions shall be interpreted as shown in Table 2.2.7, and either the GECOS Master or GECOS Slave Execution Mode may be utilized as described in the 655/EIS Processor EPS-1 and Table 2.2.7.

2.2.2 MULTICS Operational Modes

The MULTICS Operational Modes shall be in effect when the Mode Control Switch is set to the MULTICS position. In this mode, the segmentation and paging hardware of the appending unit shall be enabled and utilized as described in Section 2.3. In the MULTICS Operational Modes, the addressing mode may be either the Absolute Address Mode or the Append Address Mode as designated in Section 2.3, special instructions shall be interpreted as shown in Section 2.2.3, and the allowable execution modes shall be MULTICS Absolute, MULTICS Privileged, MULTIS Nonprivileged and MULTICS BAR as shown in Table 2.2.7.

2.2.3 Privileged Instructions

The 645F shall have three categories or privileged instructions whose attempted execution shall result in the prescribed actions specified in Table 2.2.2.

MULTICS Privileged Instructions

This class of instructions may be executed in the MULTICS Absolute and the MULTICS Privileged Modes only. Attempted execution in either of the GECOS Operational or MULTICS BAR Modes shall result in an Illegal Procesure (illegal op code) Fault, and attempted execution in the MULTICS Nonprivileged Mode shall result in an Illegal Procedure (illegal in slave) Fault. The 645F privileged instructions shall consist of:

ABSA	RCU
LDBR	SCU
LPTP	SDBR
LPTR	SSDP
LRA	SSDR
LSDP	SPTP
LSDR	SPTR

H6000 Master Mode Instructions

This class of instructions may be executed in the GECOS Master, MULTICS Absolute, and the MULTICS Privileged Operational Modes. Attempted execution of this class of instructions in the GECOS Slave Operational Mode shall result in the command fault and attempted execution in the MULTICS Nonprivileged or MULTICS BAR Operational Modes shall result in an Illegal Procedure (illegal slave procedure) Fault except the LBAR instruction shall be executable in the MULTICS Nonprivileged Mode.

The H6000 Master Mode instructions are identical to those listed in the 655 EPS-1 and consist of:

CIOC	LDT	SMCM
DIS	RMCM	SMIC
LBAR	RSCR	SSCR
LCPR	SCPR	

MULTICS Nonprivileged Instructions

This class of instructions may be executed in the MULTICS Absolute, MULTICS Privileged or the MULTICS Nonprivileged Mode. Attempted execution in either of the GECOS Operational Modes or the MULTICS BAR Mode shall result in an Illegal Procedure (illegal op code) Fault. This class of instructions shall consist of all of the new instructions specified in Section 3.4 (except MME2, MME3, MME4 and the MULTICS Privileged instructions) and consist of:

ADWPn	EPBPn	RTCD	SRA
CALL	EPPn	SPBP	STAC
EPAQ	LPRI	SPRI	STACQ
EASPn	LPRPn	SPRIn	STCD
EAWPn	RCCL	SPRPn	TSPn

2.2.4 Address Preparation

In the GECOS Operational Modes, the 645F Processor shall operate in the Absolute Address Mode only and the appending unit shall be disabled at all times (see 2.3.7 and 2.3.9.1). In the MULTICS Operational Modes the address preparation mode may be either the Absolute Address Mode or the Append Address Mode (see 2.3.7 through 2.3.9).

2.2.5 Fault Selection

In the GECOS Operational Modes, the 645F Processor shall recognize faults and interrupts as described in the 655 EPS-1. In the MULTICS Operational Modes, the 645F Processor shall recognize the extended fault set described in Table 3.7.

2.2.6 Interrupt and Fault Handling

In the GECOS Operational Modes all faults and interrupts shall be initiated and executed as presently specified in the 655 EPS-1.

In the MULTICS Operational Modes, the 645F Processor shall be able to recover from a directed fault or access violation occurring at any point during an instruction execution through the use of the SCU and RCU instructions. In addition the interrupt inhibit bit (instruction bit 28) shall be interpreted as follows:

1. In the MULTICS Absolute, MULTICS Privileged and MULTICS BAR Modes, instruction bit 28 shall inhibit external faults (connect, timer runout and shutdown) and interrupts.

2. In the MULTICS Nonprivileged Mode, instruction bit 28 shall have no effect.

2.2.7 Processor Mode Determination and Alteration

In the GECOS Operational Modes, the current machine mode (Master or Slave) and the conditions for changing modes shall be as defined in the 655 EPS-1. The absolute indicator will always be a one, however, the instructions which store it (STI, STC_1) shall store a zero in bit 31 of the indicator field and the instructions which load the indicators (LDI, RET) shall ignore bit 31 of the indicator data field.

In the MULTICS Operational Mode the particular mode of execution is determined by the Absolute (ABS) and Master Mode (M/M) indicators as shown in Table 2.2.7.

Absolute Mode

When a fault or interrupt is recognized, the Absolute Address Mode shall be used to generate the vector address and to prepare the addresses for operands of the instructions in the vector which do not have bit 29 equal to a one and do not encounter an ITS or ITP modifier or an indirect word. In addition the Absolute Address Mode shall be retained following the execution of the vector instruction pair if:

1. Either instruction is transfer of control (except TSS, RTCD, RCU or CALL) which does not have bit 29 equal to a one and does not encounter an ITS or ITP modifier.

2. Neither instruction is a transfer of control and the addressing mode prior to the fault or interrupt was the Absolute Address Mode.

3. Either instruction is an RCU or RET whose operand field specifies the Absolute Address Mode.

The only method of entering the Absolute Address Mode from the Append Address Mode other than the fault or interrupt process described above, shall be through the use of a RCU instruction executed in the MULTICS Privileged Mode whose operand field specifies the resetting of the absolute indicator.

MULTICS Append Modes

The Append Address Mode may be entered from the Absolute Address Mode only in the MULTICS Operational Mode if one of the following sets of conditions are met:

1. A TSS, RTCD or CALL instruction is executed (see Section 2.5).

2. The RCU instruction is executed and its operand field specifies the Append Address Mode.

3. Any 655 EIS transfer of control instruction (TRA, TSXn, TSPn, etc.) is executed and this instruction has bit 29 equal to a one or an ITS or ITP modifier is encountered.

4. A fault or interrupt vector pair is executed and the requirements for retaining Absolute Address Mode following their execution are not met.

In addition to the above methods of entering the Append Address Mode (and remaining there) as specified above, the Append Address Mode may be entered from the Absolute Address Mode for the execution of a single instruction if that instruction is a nontransfer instruction which has bit 29 equal to "1" or encounters an ITS or ITP modifier in an indirect word.

MULTICS BAR Mode

In addition to the specifications in Table 2.2.7, the following specifications shall enable the processor to execute GECOS Slave programs in the MULTICS BAR Mode.

1. Bits 0-2, 29 of an instruction shall be recognized by the control unit for selecting an Address Register modifier (ADRn) but the appending unit shall not select an external segment number (all memory accesses shall be to the current procedure segment) or change the ring number (level of privilege) of the current procedure segment (see 2.3.3 and 2.4.2).

When the processor is in the MULTICS Operational Mode there are two ways of entering the MULTICS BAR Mode of execution.

1. Execution of the TSS instruction (see 2.5.5).

2. Execution of a RCU instruction whose operand field specifies MULTICS BAR Mode.

Table 2.2.7 - 645F Modes

	GECOS OPERATIONAL		MULTICS OPERATIONAL			
	GECOS Master	GECOS Slave	MULTICS Absolute	MULTICS Privileged	MULTICS Nonprivileged	MULTICS BAR
Address Mode	Absolute			Append		
Absolute Indicator (ABS)	*2 1	*2 1	1	0	0	0
Master Mode Indicator	1	0	*3	*3	*3	*3
$\overline{\text{BAR}}$ Mode Indicator	*3	*3	1	1	1	0
PPR.P *1	*2	*2	1	1	0	1 or 0
H6000 M/M Instructions	OK	CMD FLT	OK	OK	IPR FLT Priv. Inst.	IPR FLT Priv. Inst.
MULTICS Privileged Instructions	IPR FLT (IOC)	IPR FLT (IOC)	OK	OK	IPR FLT Priv. Inst.	IPR FLT (IOC)
MULTICS Nonprivileged Instructions	IPR FLT (IOC)	IPR FLT (IOC)	OK	OK	OK	IPR FLT (IOC)
TSS Instructions	OK	TRA	OK	OK	OK	TRA
ITS Modifier	IPR FLT	IPR FLT	OK	OK	OK	IPR FLT
Instruction Bit 28	Inhibits	Inhibits	Inhibits	Inhibits	Ignored	Inhibits
LBAR	OK	CMD FLT	OK	OK	OK	IPR FLT Priv. Inst.

*1 Procedure Pointer Register Assembly - privileged bit (see 2.3.6.5).

*2 Ignored - that is, stored as zero and not loaded to a zero.

*3 When operating in the GECOS Mode the indicator is defined as Master/Slave; while in the MULTICS Mode it is defined as the $\overline{\text{BAR}}$/BAR indicator.

Once the processor has entered the MULTICS BAR Mode, it shall remain in that mode until a fault or interrupt is initiated. The processor will re-enter the MULTICS BAR Mode following the execution of the vector instructions if:

1. Neither instruction is a transfer of control.

2. Either instruction is a TSS or RCU which would normally cause the MULTICS BAR Mode to be entered.

2.3 ADDRESS PREPARATION AND MODIFICATION

The concepts and specifications for segmentation and paging in the 645F system shall be similar to those of the 645 Processor. Changes will be made only where MULTICS software experience has shown that significant cost/performance enhancements are to be gained. The general philosophy of segmentation and paging will be stated here as an introduction to the appending unit address preparation hardware.

2.3.1 Terms and Definitions

The following terms and definitions peculiar to segmentation are listed to assist the reader in understanding subsequent discussions and descriptions presented in later paragraphs and sections of this EPS.

Absolute Base Address - the core store address of the first word of the page or segment being referenced.

Appending - uniting and/or linking specific combinations of segment number registers, segment descriptor words, page table words and computed address to form an absolute memory address. Also, the uniting of relative address to base addresses to develop effective addresses and pointers. The term appending also infers a

simultaneous verification that the address is within limits and that other access criteria are met.

Associative Storage - high-speed storage used in appending which contains recently referenced segment and page locations and other related information. (See 2.3.6 for word formats.)

Descriptor Segment - a list of words (SDW's) which is used as an index to point to segments or to segment page tables, and describes the size of such segments and the restrictions on their access.

Target Segment/Page - refers to the segment or page containing a desired instruction, operand or indirect word to be read or stored into.

Effective Segment Number (ESN) - refers to the relative descriptor segment page and/or word number address generated by the appending unit.

Computed Address (CA) - refers to the relative object segment page and/or word number address generated by the control unit. This address is prepared as specified in EPS-1 for the 6000 Extended Instruction Set Processor. The computed address provides a logically sequential address to the user whereas the absolute memory address used to access main storage may be quite different.

Final Address (FA) - any memory address which does not access a SDW, PTW or DSPTW.

Effective Address(EA) - a computed address which does not access an indirect word.

Procedure Segment - a segment which consists primarily of instructions. A procedure segment is considered "pure" if it is unaltered by execution.

Current Procedure Segment - the segment currently referenced by the procedure pointer register from which sequential instructions are executed.

Process - a sequence of actions (programs) comprised of procedure and data segments to be executed by the processor. The sequence may or may not be altered during execution.

Segment - an ordered set of words referenced by a unique name. The segment may consist of instructions and/or data words. It has a fixed length. The entire segment or portions of it may reside in storage during execution. The maximum addressable segment length is 256K.

Segment Descriptor Word (SDW) - a word which contains the address of either an object segment or an object segment page table in main storage, a segment size field and segment access information.

Page Table Word(PTW) - a word which contains the absolute base address of a descriptor segment page (DSPTW) or the absolute base address of an object segment page. The PTW also contains a field to indicate page usage and modification information.

Segment Tag - this tag, when used, consists of the three most significant bits of the instruction word. It specifies one of eight pointer registers to be used in appending.

Page - a fixed subdivision of continuous address within a segment. The page size of the 645F Processor will be a single hardwired value (initially one thousand twenty-four (1024) word). (See 2.3.6.3.) The maximum number of pages within a segment is equal to $\frac{256K}{\#\text{ words per page}}$.

Page Table - a continuous set of words in memory which contain the Page Table Words (PTW) for a segment.

2.3.2 Segmentation Concepts

Segmentation refers to the concept of dividing a program into self-contained parts in order to be able to execute the program without maintaining it in its entirety in storage at any one time.

A segment is a collection of instructions and/or data which are all associated with a particular segment name.

A subroutine, such as a square root subroutine, can be written as a segment. Now many users can incorporate the subroutine in their process by specifying the segment name. The user does not need to describe or copy the individual instructions. Similarly, data may be shared among many users by each user specifying the name of the data segment.

Occasionally, a segment will be found to have an error in it. Or, a need will arise for having the segment do more or do things in a better way. The segment is changed and recompiled or reassembled. Usually, the new version will perform the same functions as the old version and with the same interface to the user. When this is the case, the program which used the old segment needs no changes in order to use the new segment.

Since segments are relatively independent of each other, many people can be preparing segments at the same time. Symbolic names can be used freely. Obviously, when an instruction in segment A refers to a word in segment B, the segment A instruction must know the name of segment B and the name of the desired word in segment B. The important point is that the choice of names for the words in segment B does not need to be coordinated with the writers of other segments, except for those names referred to by other segments.

Communication between segments is done via symbolic names. Therefore, a segment does not have to be in a specific storage location.

There is no requirement for all the segments which belong to a process to be in main storage at the same time. Hence, segments which are not being used, do not occupy valuable main storage space but reside in secondary storage such as a drum or disk or tape.

Another way to save storage is to use pure procedure segments. These are segments which do not modify themselves. They are written in a re-entrant, recursive manner. Therefore, all programs can use the same copy of a pure procedure segment. Thus, only one copy of the segment is needed in storage, rather than one copy for every program which uses that segment.

The major reasons for segmentation are as follows:

1. The user is able to program in an essentially infinite storage system. Any one segment can contain up to a quarter million words. On the 645F Processor, any single segment can dynamically grow or shrink between the limits of 16 to 256K words.

2. The user can operate his program through phases of segment configuration without prior planning of the storage allocation needs of the management of the segments.

3. The largest amount of code which must be bound together as a solid block is a single segment. Since binding pieces of code together is a process which is similar to assembly or compiling, the user is able to prepare arbitrarily large programs out of a series of segments connected by an efficient binding method.

4. Allows shoring of procedure and data bases.

5. User programs and data can be protected by the access restrictions placed on individual segments.

2.3.3 Segmentation Implementation in the 645F

In the Append Address Mode, main memory is divided into segments each of which is described by a Segment Descriptor Word (SDW). When a process is placed in execution by MULTICS, the SDWs for this process are placed in the descriptor segment. The Descriptor Segment Base Register (DSBR) is then loaded to specify the absolute base address of this descriptor segment (DSBR.ADDR) and the number of 16 word blocks of SDWs (DSBR.BND) in the descriptor segment.

2.3.3.1 Current Procedure Segment Accessing

When a process is in execution, the segment from which it is executing instructions is called the current procedure segment. This segment shall be described by the Procedure Pointer Register (see 2.3.6.5) which designates a level of privilege or ring number (PPR.PRR 00-02), the segment number (PPR.PSR 00-14) and relative address within the segment of the current instruction being executed (PPR.IC 00-17). Figure 2.3.3.1a depicts instruction fetching from the procedure segment. All sequential instruction fetches and all operand accesses which do not have instruction bit 29 equal to a one and do not encounter an ITS/ITP modifier in an indirect word, will be directed to the procedure segment as shown in Figure 2.3.3.1a. The only difference between instruction and operand fetches from the current procedure segment shall be in the generation of the Temporary Pointer Register - Computed Address (TPR.CA). A full description of TPR.CA is given in Section 2.3.9.1.

The Procedure Pointer Register shall be altered by transfer of control instructions which utilize bit 29 or an ITS modifier in an indirect word (see 2.3.3.2 and 2.3.3.3) and by the RTCD, CALL and RCU instructions as described in Section 2.5. Figure 2.3.3.1b illustrates how the TRA instruction can be used to establish a new current procedure segment.

2.3.3.2 Direct Segment Accessing

In the 645F Processor, the current procedure segment shall be able to directly access eight segments other than itself. This shall be accomplished by using eight Pointer Registers (PRn) which contain a segment number (PRn.SNR) a word offset (PRn.ADR) and a level of privilege (PRn.PNR). These Pointer Registers shall be used whenever bit 29 of an instruction is a one. The specific Pointer Register shall be specified by bits 00-02 of the address field (y) of the instruction and bit 03 of (y) shall specify whether y is a positive ($y_{03} = 0$) or negative ($y_{03} = 1$). Figure 2.3.3.2 illustrates the accessing of an external segment using Pointer Register 2.

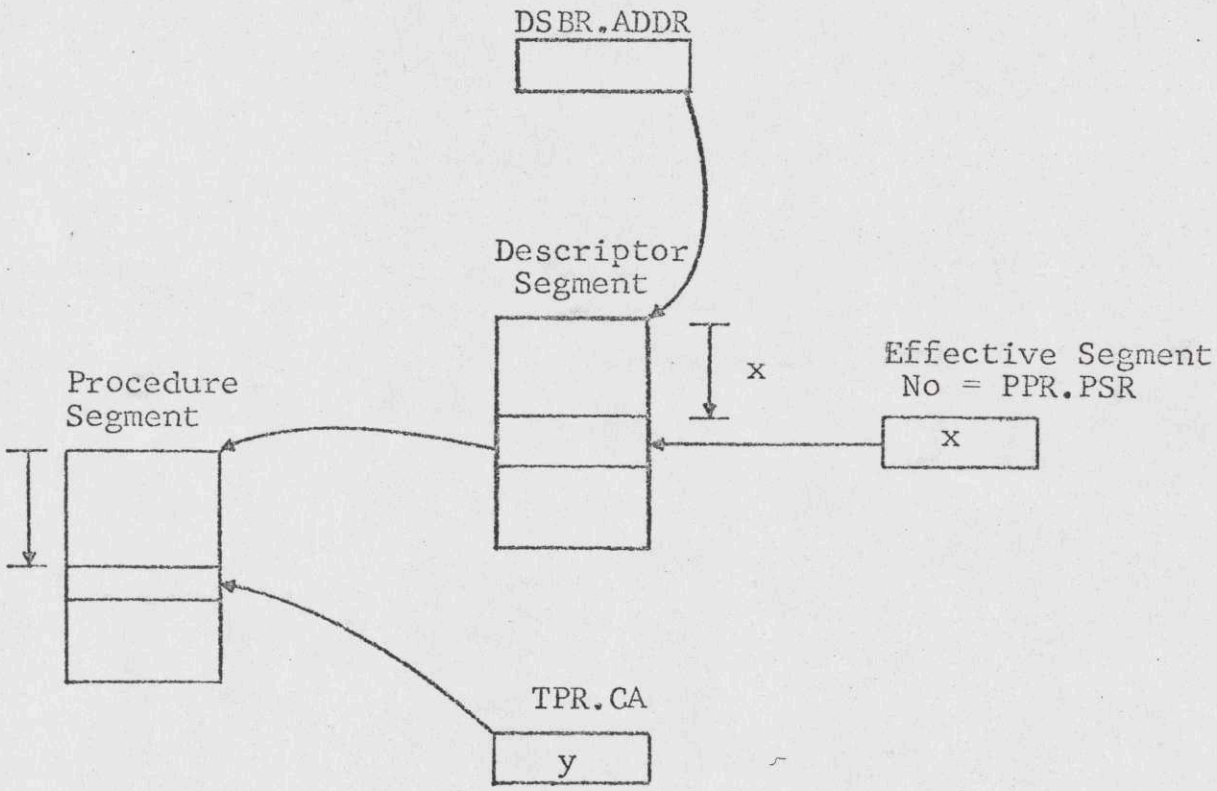


Figure 2.3.3.1a - PPR.PSR Designates the Segment from which Instructions are Fetched

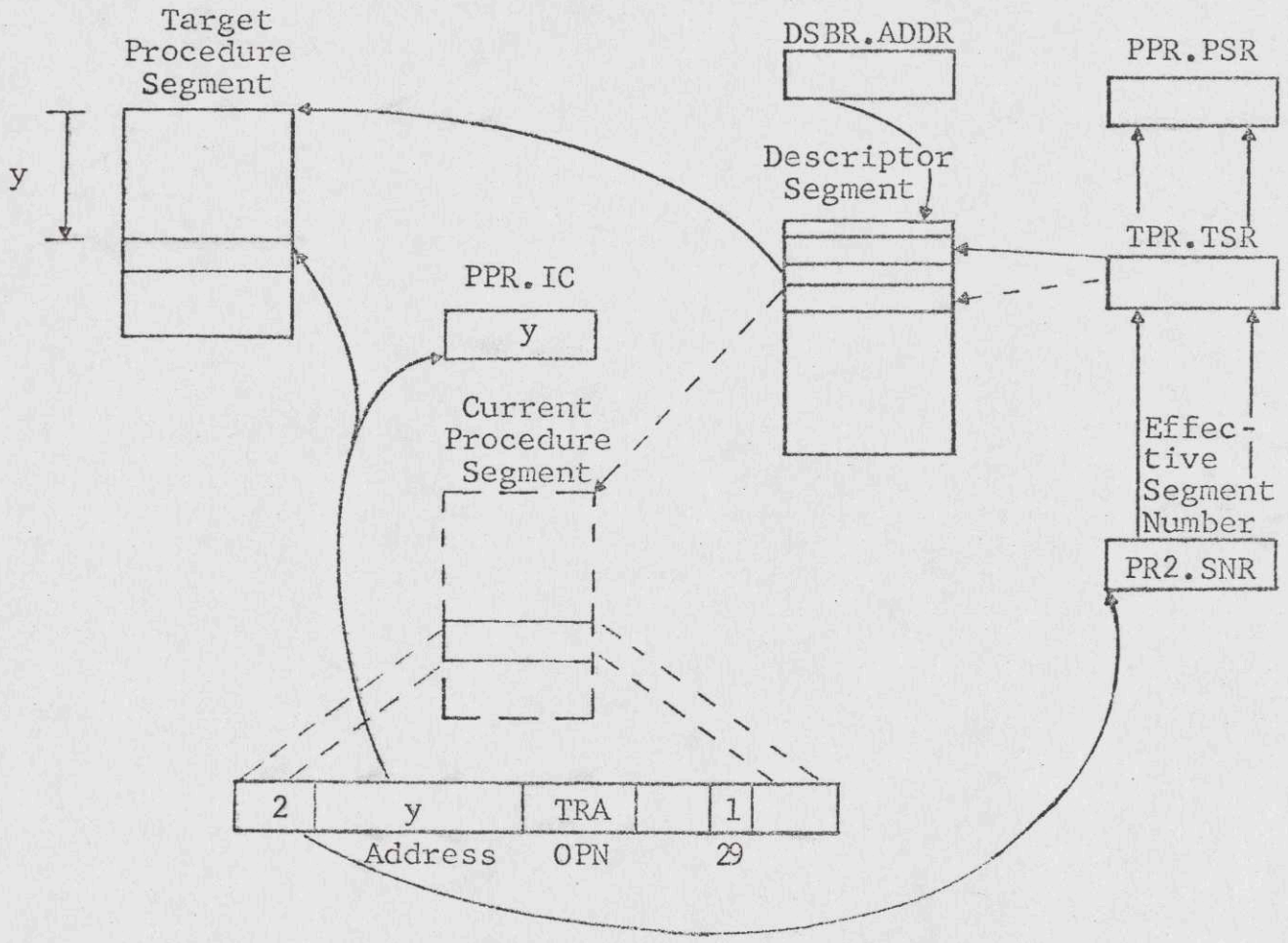


Figure 2.3.3.1b - A Transfer Instruction causes the Procedure Segment Register to be Changed

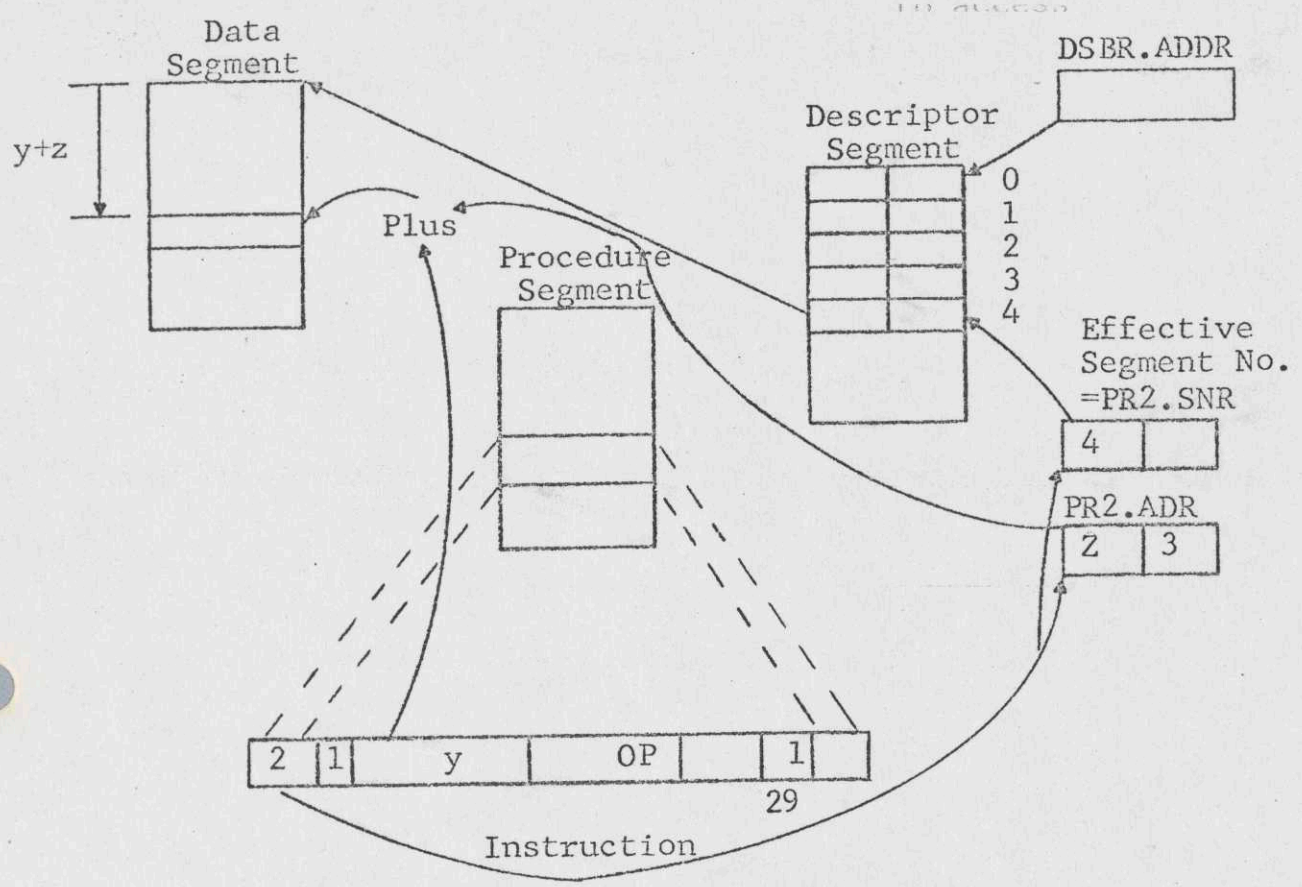


Figure 2.3.3.2 - A Pointer Register is used for Direct Reference to Another Segment

2.3.3.3 Indirect Segment Accessing

In addition to the capability to access eight segments external to the current procedure segment, the 645F Processor shall provide the capability to indirectly access any segment which is described in the SDW Table of the descriptor segment through the indirect to segment (ITS) modifier or the indirect to pointer (ITP). The 645F Processor shall access two words, in anticipation of the ITS or ITP modifier, any time an instruction calls for RI or IR modification to an even memory address in the 645 Operational Mode.

Upon receipt of the indirect word pair, the processor shall examine the even word for the presence of the ITS or ITP modifier tag. If the ITS or ITP tag is not present, the first word shall be treated as a regular indirect word. If the ITS tag is present, the appending unit shall use the address field (bits 03-17) of the even word as a new effective segment number and shall treat the odd word as a regular indirect word. Figure 2.3.3.3 illustrates the application of the ITS modifier and Section 2.5.7 describes the ITS modifier in detail.

If the ITP tag is present, the appending unit shall use bits 00-02 of the even word to designate a Pointer Register to be used in the formation of the address of the operand or next indirect word and the odd word of the pair shall be treated as a regular indirect word. Figure 2.3.3.4 illustrates the application of the ITP modifier and Section 2.5.8 describes the ITP modifier in detail.

2.3.4 Paging Concepts

Each segment does not always have to occupy fixed storage locations. Nor do all segments for a program have to be in storage at the same time. Paging carries this concept even further, in that not all of the pages which comprise a segment need be in storage at the same time.

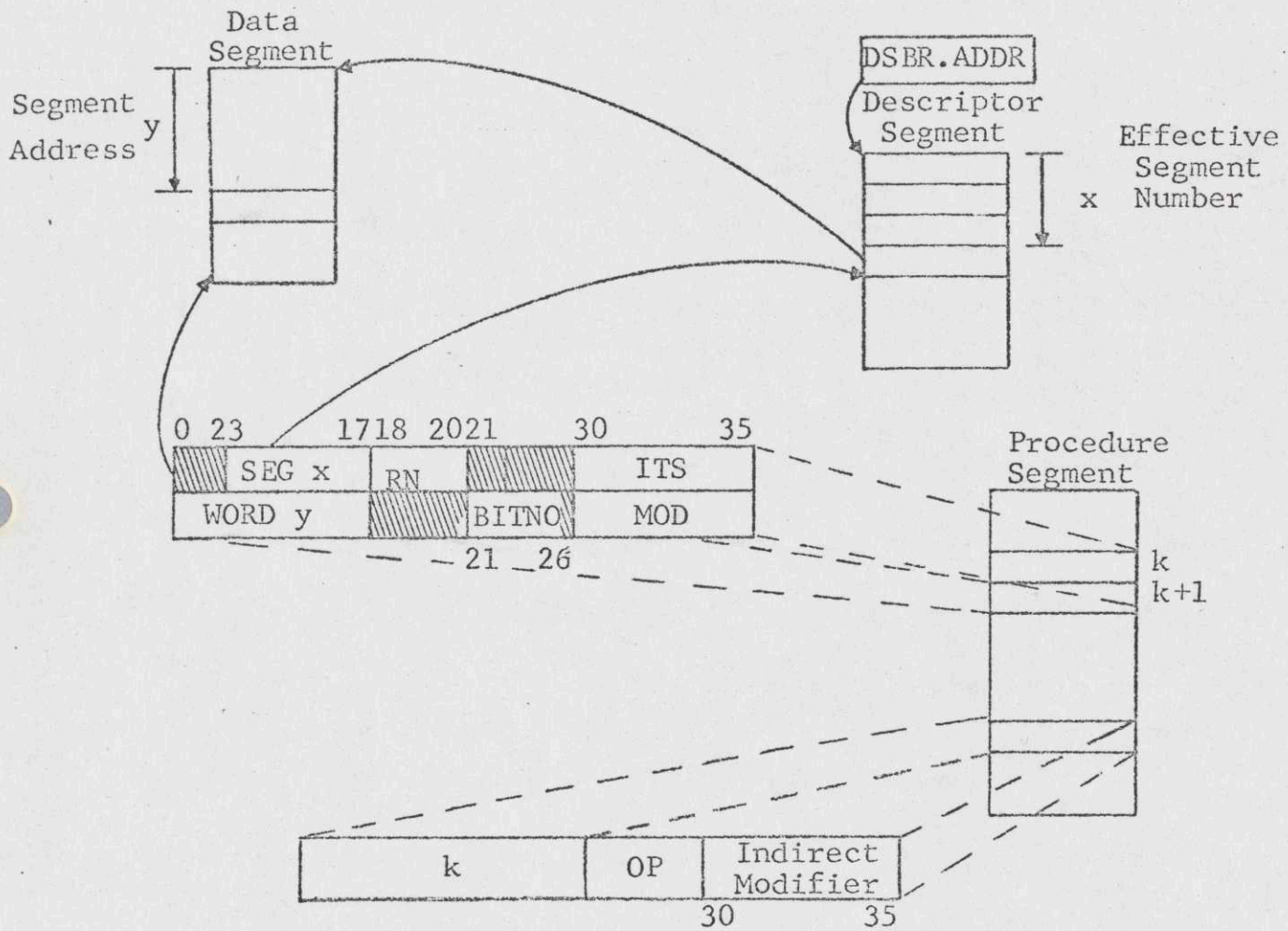


Figure 2.3.3.3 - An ITS Pair is Used to Refer to Another Segment

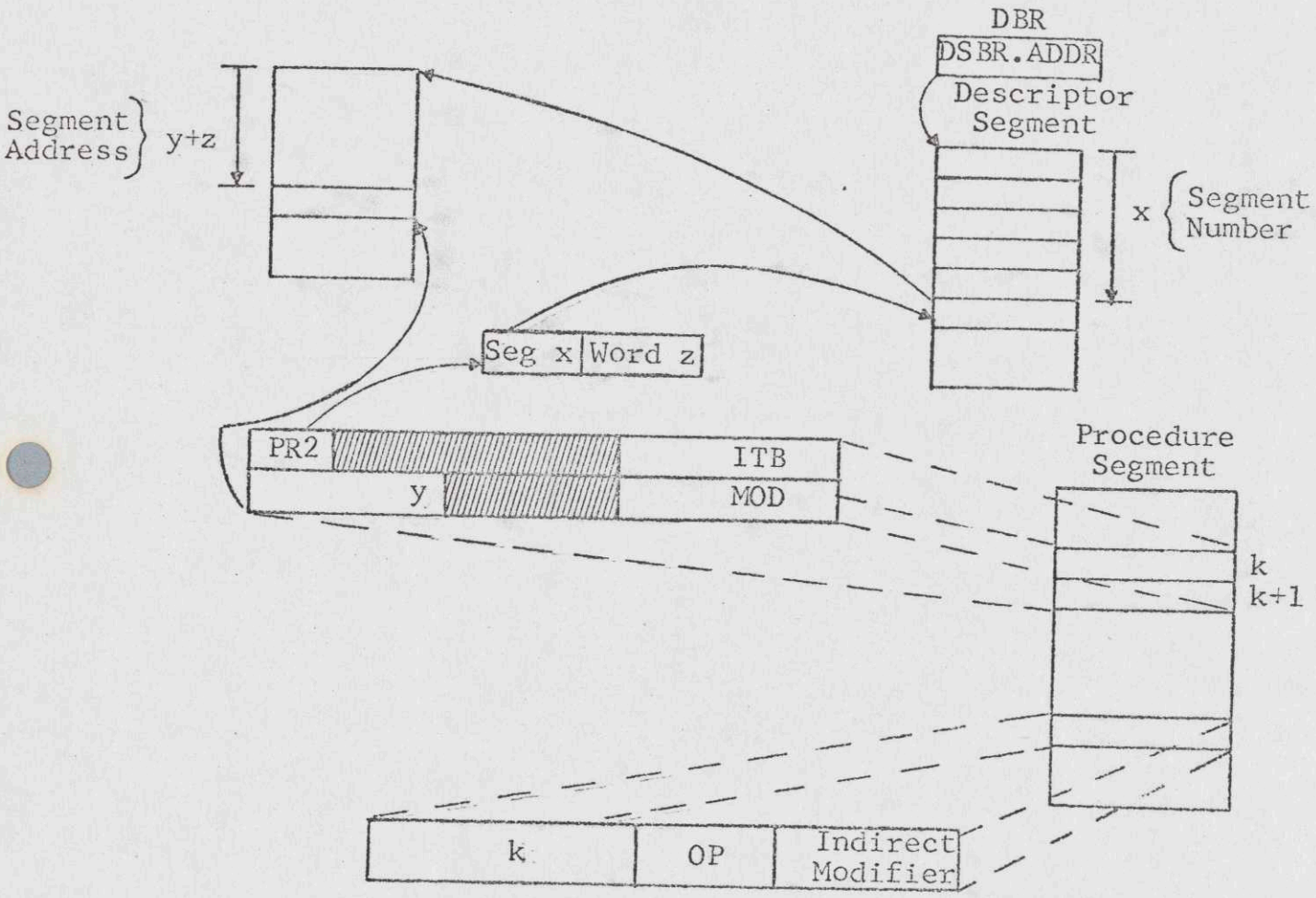


Figure 2.3.3.4 - An ITP Pair is Used to Refer to Another Segment

Assume for a moment that an entire segment is placed in storage. Now think of the storage occupied by this segment as being partitioned into several blocks, all the same size. The segment can be divided into several pages, each of which contains the same number of words as the storage block size.

The essential point of the paging concept is that the pages of a given segment need not be located in contiguous storage blocks.

As pages are needed in main memory they are retrieved from secondary storage. The pages are placed in any available block of main store. These pages are described by Page Table Words (PTWs) which specify the Absolute Base Address (PTW.ADDR) of the page in main memory in addition to control information about the page. These PTWs are located in a page table which specifies all the pages for a given segment (see Section 2.3.6.3).

Eventually, the main memory will be full and more pages will be needed. Some pages are going to have to be displaced. However, if a page which is no longer needed is part of a pure procedure segment, or if the page has never been written into, the operating system does not have to swap out the page since it is still in the secondary storage.

Any access controls which were established for a segment obviously apply to the pages which make up that segment.

Paging has two distinct advantages:

1. It allows a flexible technique of dynamic storage management and reduces operating system overhead which is especially high in heavy-traffic situations such as occur in responsive time-sharing systems.

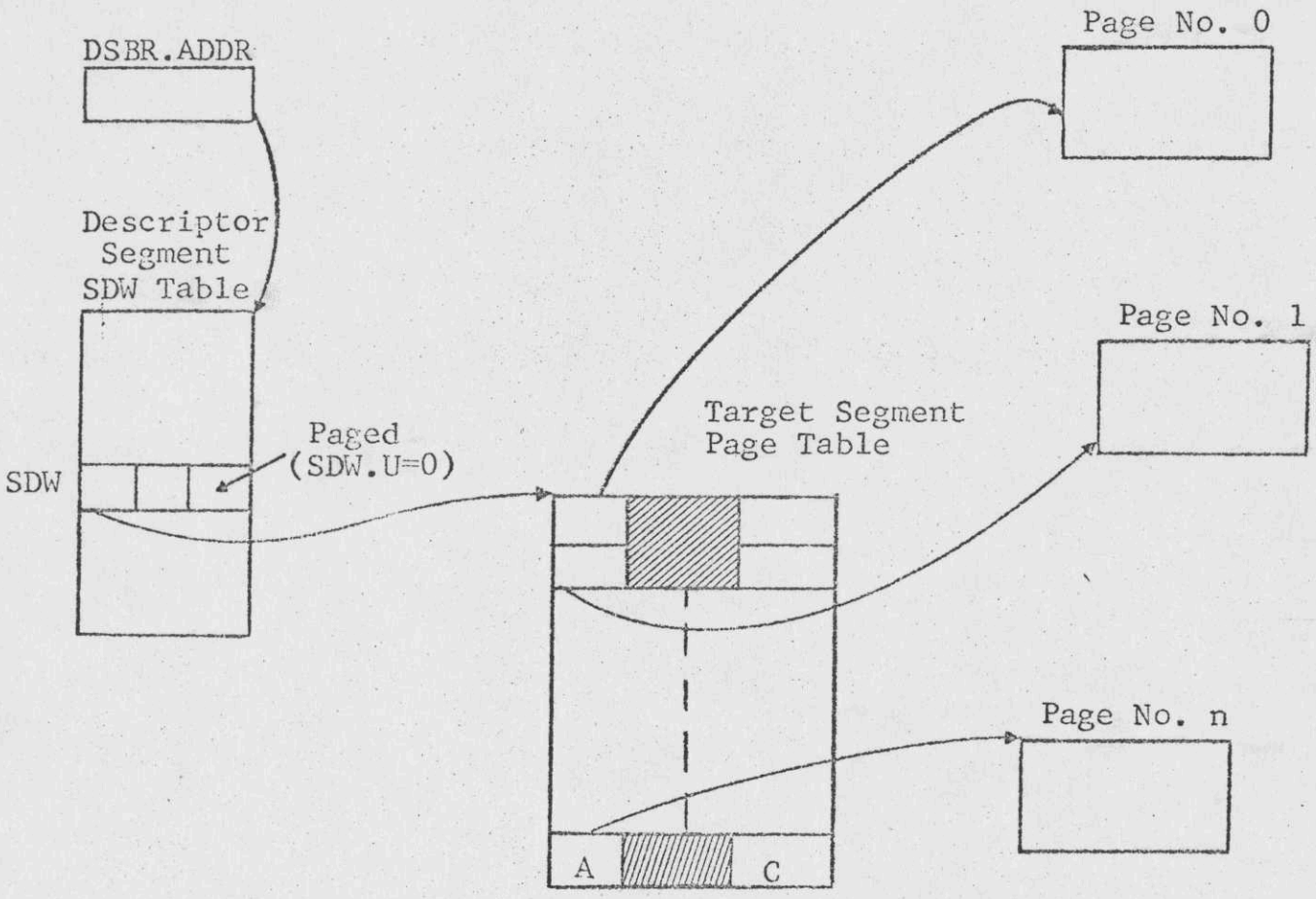
2. The mechanism of paging enhances system operation through effective utilization of high-speed storage.

2.3.5 Paging Implementation in the 645F

The paging scheme which shall be implemented in the 645F Processor shall be very similar to the 645 Processor paging scheme. The major exception shall be that the 645F Processor shall implement a single page size of 1024 words which may be alterable between 64 words and 4096 words by Field Engineer personnel. (See 2.3.6.1.)

2.3.5.1 Paged Object Segments

In the Append Address Mode, any segment (including a descriptor segment) may be paged. The determination as to whether or not a procedure or data segment is paged shall be made by examination of the unpaged control bit (SDW.U) in an SDW when it is accessed. If SDW.U is a one then the segment shall be unpaged. If SDW.U is a zero then the segment shall be paged and the address field of the SDW (SDW.ADDR) shall be interpreted as the base address of the Segment Page Table rather than the segment itself. In addition, for paged segments the computed address (CA), which is the relative address generated by the control unit, shall be interpreted as the page number and word number of the instruction operand, etc. rather than just the word number (see Section 2.3.9.1). In this format the page number portion of the computed address (CA.PG) shall specify the PTW number (address relative to the base of the page table) and word portion (CA.WD) shall specify the relative address of the operand, instruction,etc. in the object page whose base address is specified by the PTW for the page. Figures 2.3.51a and 2.3.5.1b illustrate the basic paging concepts which shall be implemented for paged object segments in the 645F Processor.



A = Page Origin
B = Control Information

Figure 2.3.5.1a - Page Table Words Specify Absolute Core Memory Origins

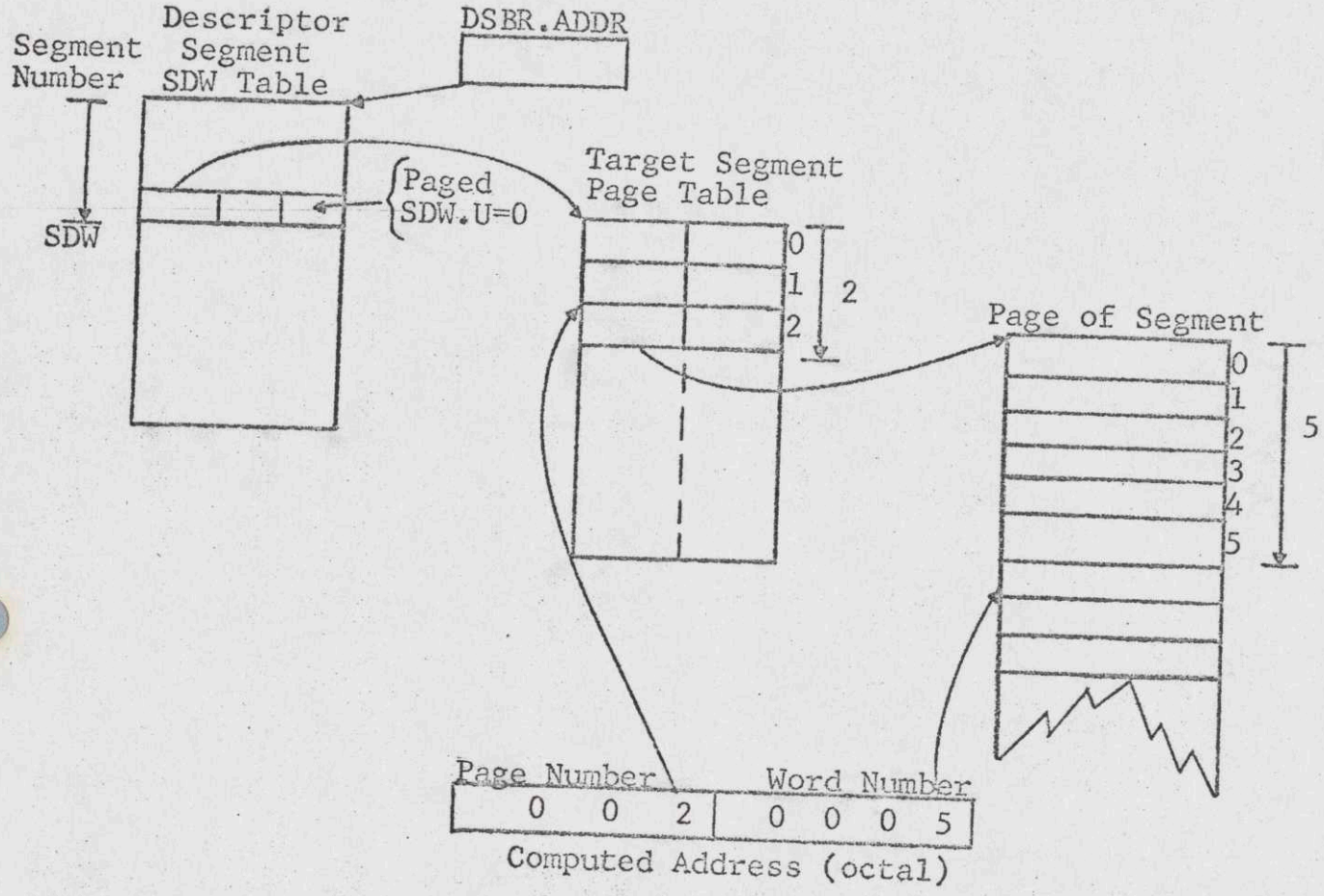


Figure 2.3.5.1b - Selection of a Word From a Paged Segment

2.3.5.2 Paged Descriptor Segments

As previously mentioned, the 645F Processor shall provide the capability to specify paged descriptor segments. This function shall be accomplished by providing an unpaged control bit in the Descriptor Segment Base Register (DSBR.U). When DSBR.U is a zero the descriptor segment for the processes in execution shall be paged and the address field of the DSBR (DSBR.ADDR) shall be used to specify the base address of the Descriptor Segment Page Table and the effective segment number shall be interpreted as page and word reference for the descriptor segment. The page number field of the effective segment number (ESN.PG) shall specify the address of the object segment in the descriptor segment page whose base address shall be specified by the DSPTW. Figures 2.3.5.2a through 2.3.5.2f show the application of paged object and descriptor segments. However, whether or not the descriptor segment is paged shall be completely independent of whether or not the object segment is paged.

2.3.6 Segmentation and Paging Hardware Requirements

2.3.6.1 Variances from the 645-Processor

The segmentation and paging principles of operation in the 645F Processor shall differ from the 645 Processor in the following respects:

1. All pages in the 645F Processor shall be the same size rather than two possible sizes as in the 645 Processor. The page size will initially be 1024 words. However, it will be possible to alter the page size in a particular machine via a field modification to 2^n words ($6 \leq n \leq 12$). Regardless of page size the maximum segment size that can be specified shall be 262,144 words (256K).

2. The 645F Processor shall include two 16 word associative memories - one for Segment Descriptor Words (SDW) and one for Page Table Words (PTW). (Section 2.3.6.8 and 2.3.6.9.)

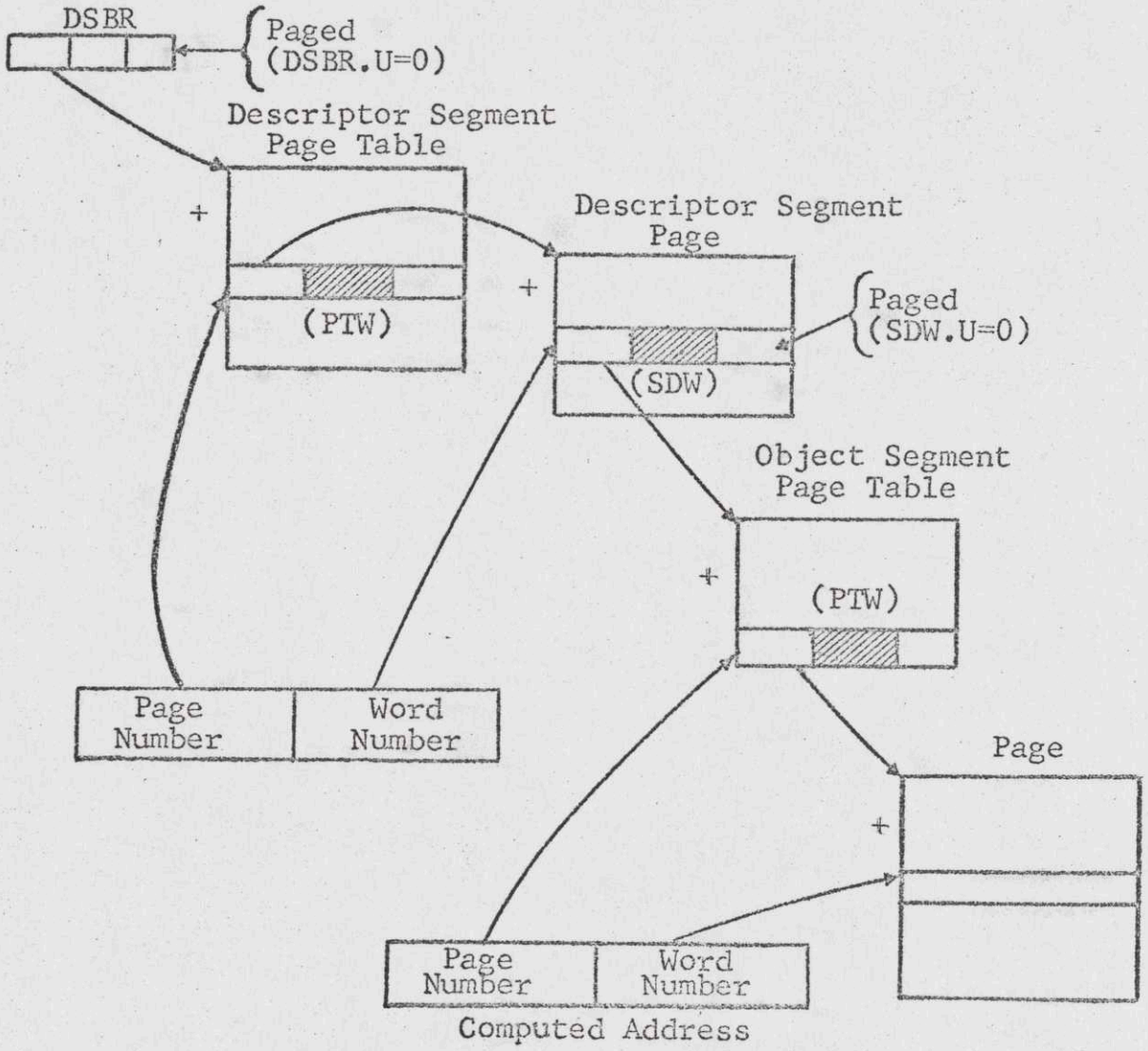


Figure 2.3.5.2a - Selection of a Word from a Paged Segment Using a Paged Descriptor Segment

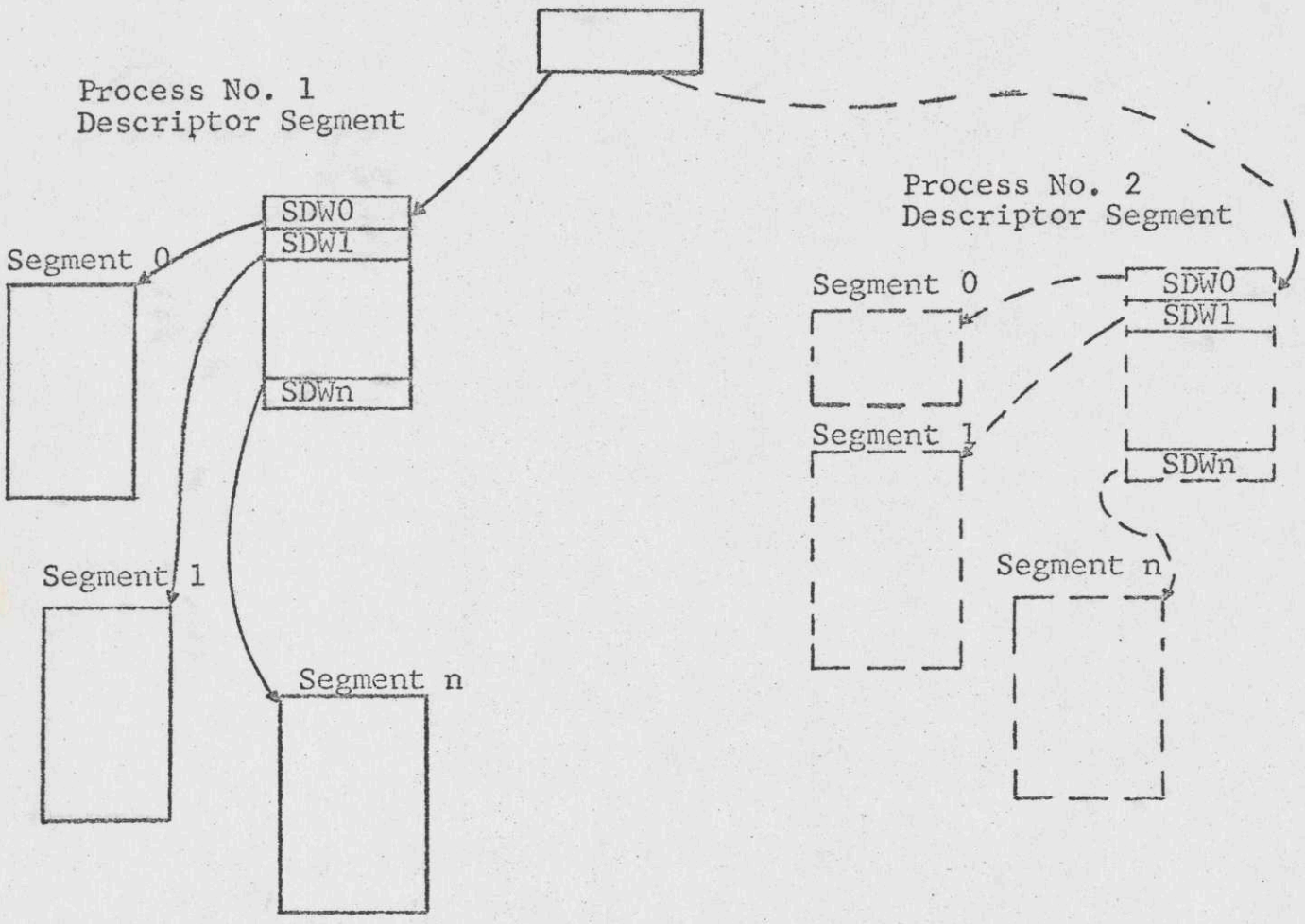


Figure 2.3.5.2b - The DSBR Determines which Segments the Processor May Access

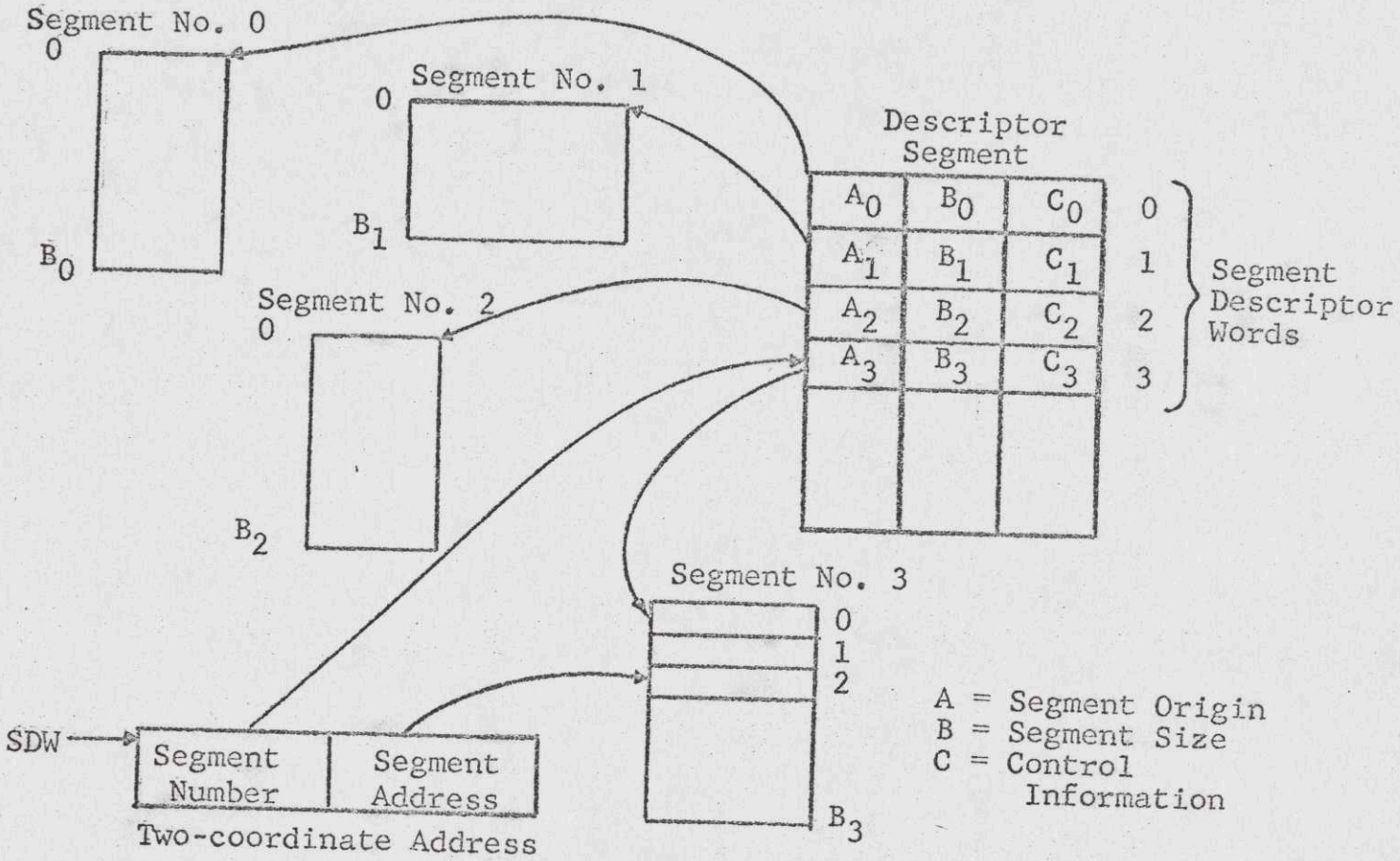


Figure 2.3.5.2c - The Descriptor Segment Facilitates Two-Coordinate Addressing

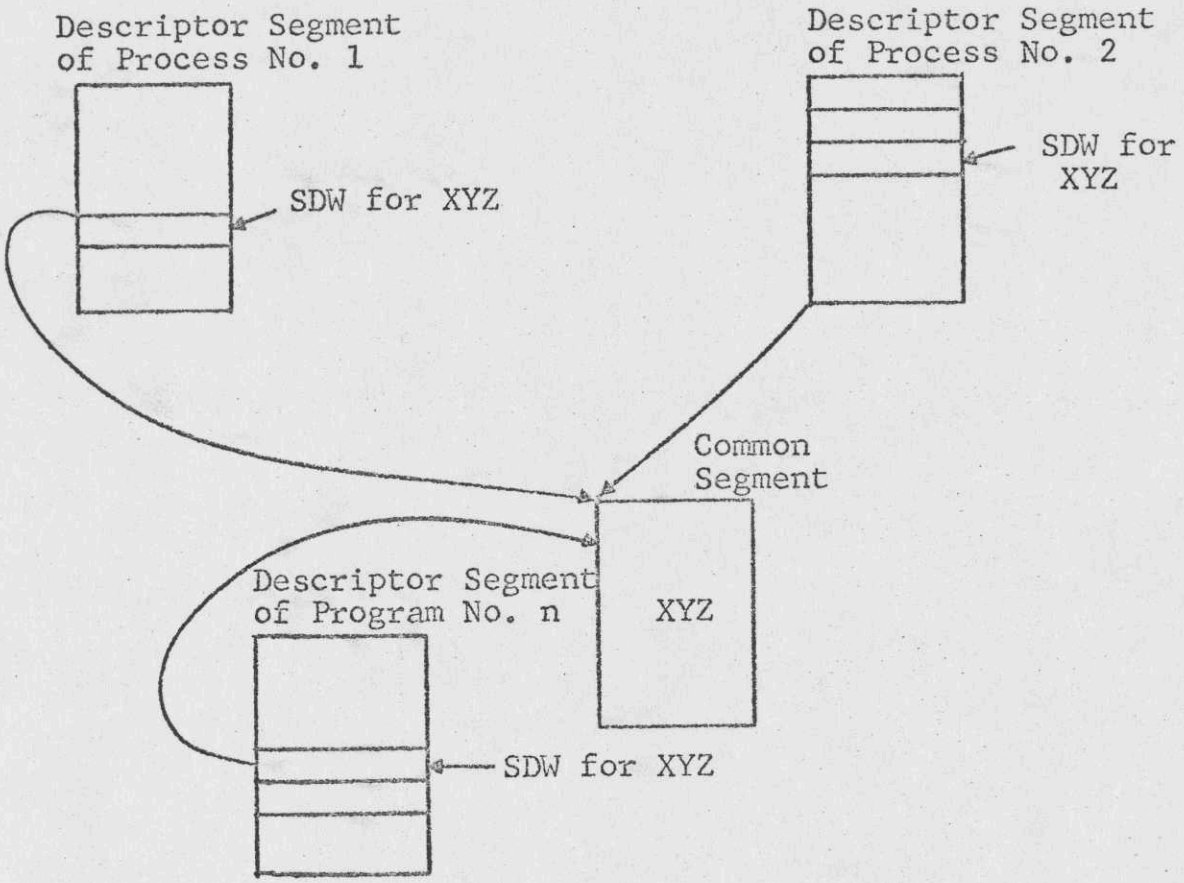


Figure 2.3.5.2d - Several Descriptor Segments May Refer to a Common Segment

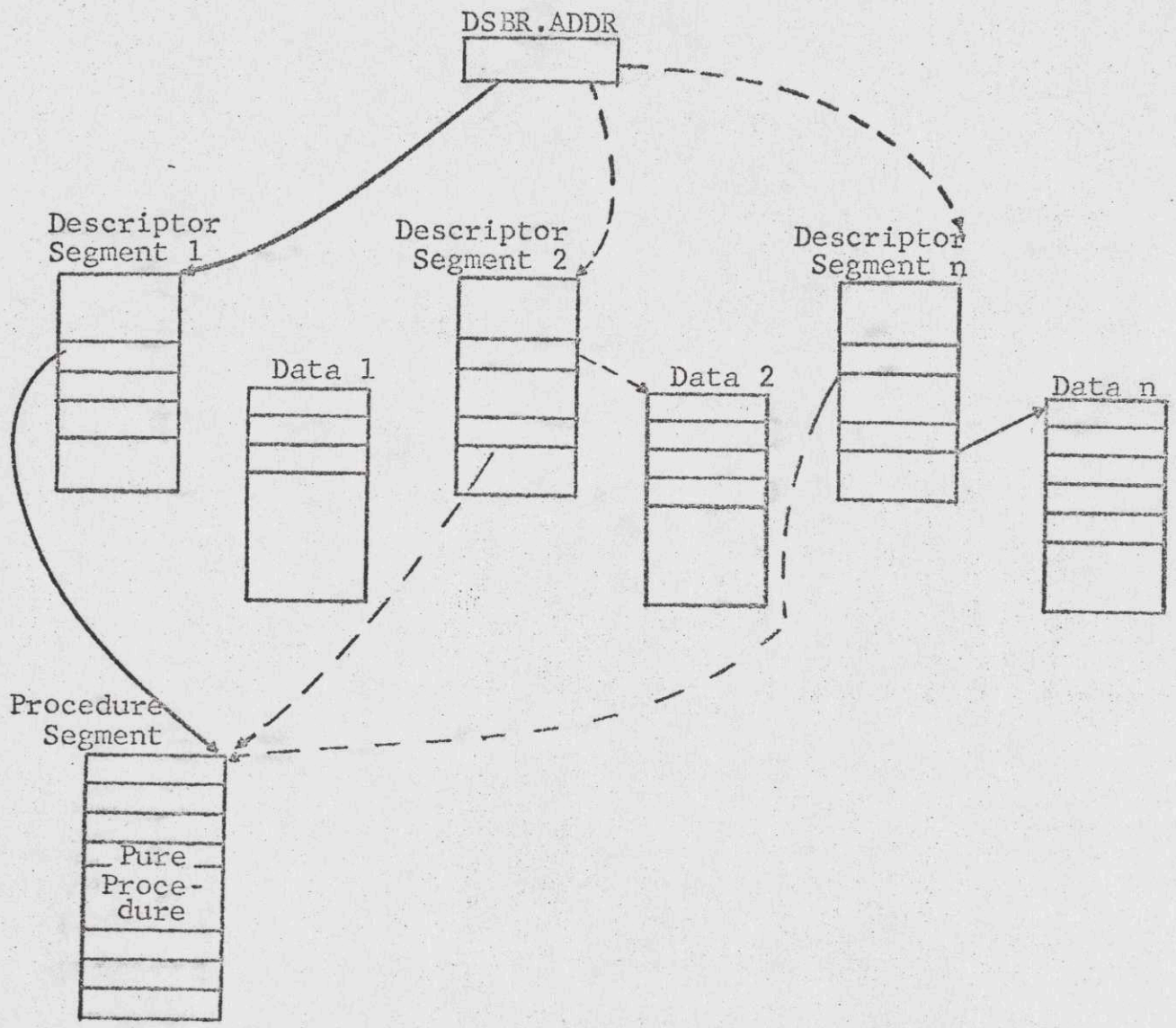


Figure 2.3.5.2e - Common Pure Procedure Alternately Serves n Programs in a Uniprocessor Configuration

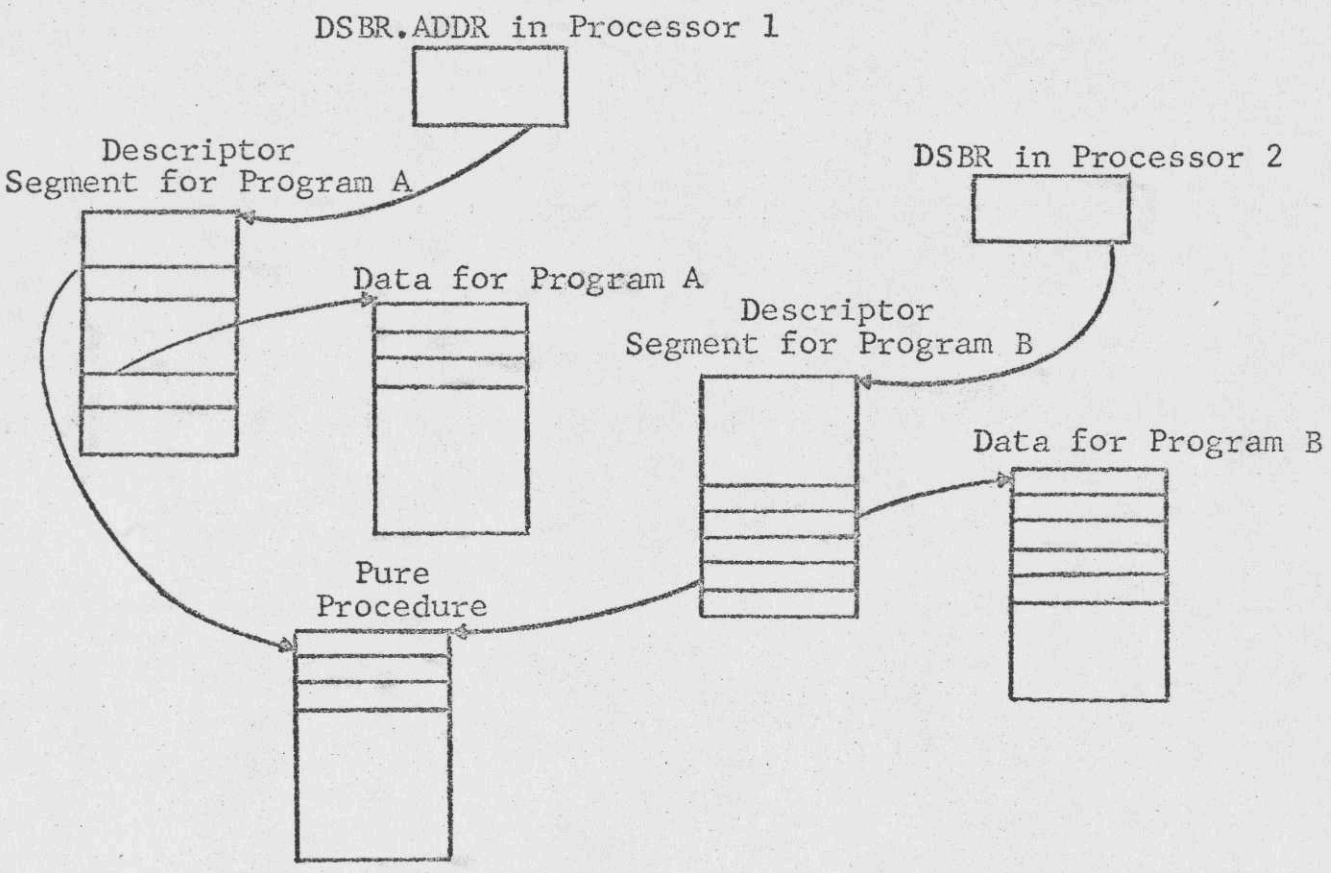


Figure 2.3.5.2f - Common Pure Procedure used Concurrently by Two Processors in Dual Configuration

3. The eight ABRs in the 645 Processor which can be used as either external or internal offset fields shall be replaced by eight segment number registers (SNRn) and the eight address registers (ADRn) of the 655 EIS Processor shall be used as internal offset registers (Section 2.3.6.7).

4. The number of directed fault types shall be reduced from eight to four in the 645F Processor.

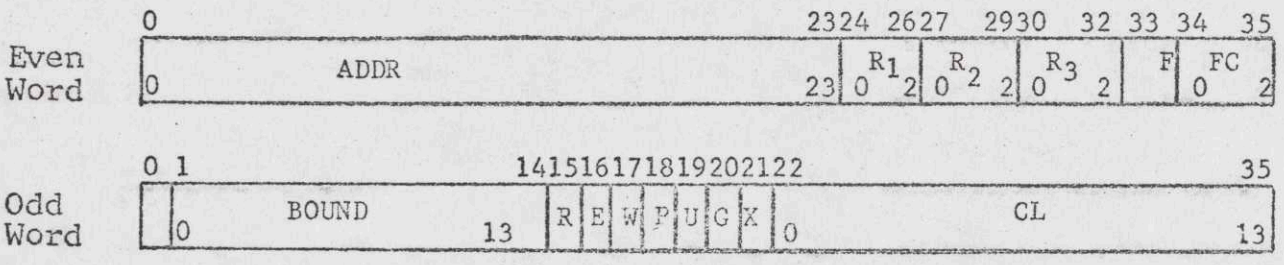
5. The segment bounds control in the 645F Processor shall check all segment addresses to the highest 16 word block which can be addressed (ref. 2.3.6.2 and 2.4.1).

6. The segment access control mechanism in the 645F Processor shall differ extensively from the 645 due to the hardware implementation of the MULTICS ring structure (Section 2.4).

7. The 645F Processor shall contain a switch on the maintenance panel which may be used to disable the appending mechanism and force the 645F Processor to function as a 6000 EIS Processor (Section 2.2).

2.3.6.2 SDW Format

In the 645F Processor, the SDW in memory shall consist of two words in the following format:



where:

ADDR is the 24 bit segment address whose use is determined by the state of control bit U (bit 19 of word 2).

R_1 specifies the highest effective ring number of the read/write bracket for this segment (Section 2.4.3).

R_2 specifies the highest effective ring number of the read/execute bracket for this segment (Section 2.4.3).

R_3 specifies the highest effective ring number of the call bracket for this segment (Section 2.4.3).

F directed fault indicator.

1 - the necessary unpaged segment or PTW currently resides in memory.

0 - execute the directed fault specified in FC.

FC indicates (if F = 0) which of the four directed faults is to be executed (DF0-DF3).

BOUND specifies the highest mod 16 computed address which can be generated without causing an out of bounds fault. This 14 bit bound field is compared to the most significant 14 bits of the computed address prior to each memory access for a PTW, instruction, indirect word or operand.

R is the read permission bit.

0 - no segment other than itself can read from this segment.

1 - any procedure segment executing from a ring less than or equal to R_2 can read from this segment.

E is the execute permission bit.

0 - a processor cannot fetch instructions from this segment (XEC, XED excluded).

1 - a processor executing in a ring greater than or equal to R_1 and less than or equal to R_2 can fetch instructions from this segment.

W is the write permission bit.

0 - no procedure segment can alter this segment.

1 - any procedure segment executing from a ring less than or equal to R_1 can alter this segment.

P is privileged mode bit. (Applies only when the SDW specifies the current procedure segment.)

0 - privileged instructions cannot be executed and interrupts cannot be inhibited via bit 28.

1 - if this procedure segment is executed in ring 0, it can execute privileged instructions and inhibit interrupts via bit 28.

U is the unpaged/paged bit (see Section 2.2.2.2).

0 - this segment is paged, $SDW.ADDR_{00-23}$ is the base address of the segment page table.

1 - this segment is unpaged, $SDW.ADDR_{00-23}$ is the base address of the segment.

G is the gate indicator.

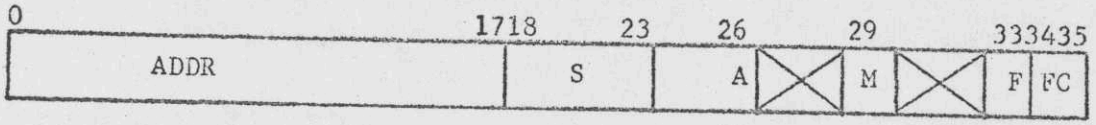
0 - any call to this segment from a different segment must be directed to a computed address less than the value of the CL field or an Illegal Procedure Fault will be generated.

1 - any legal segment address may be called by a different segment.

CL is the largest computed address in this segment that can be called by another segment if G = 0.

2.3.6.3 PTW Format

In the 645F Processor, the PTW in memory shall have the following format:



where:

ADDR is the address of a page. The hardware ignores certain lower order bits of ADDR in order to generate the proper hard-wired page size according to the following table.

Page Size (Words)	ADDR Bits Ignored
64	
128	17
256	16-17
512	15-17
1024	14-17
2048	13-17
4096	12-17

S is page status information reserved for software use (ignored by the hardware).

A is the page accessed bit which is set to a one any time the hardware accesses this PTW and finds the A bit equal to a zero.

M is the page modified bit which is set to a one the first time the hardware alters the contents of the page referenced by this PTW.

F is the directed fault indicator.

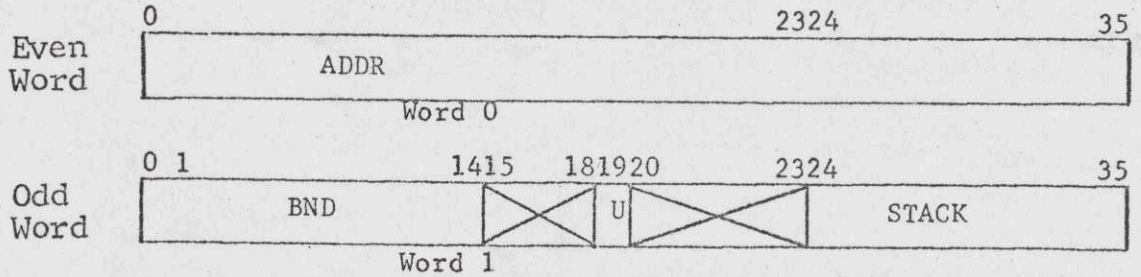
1 - the necessary page resides in main memory.

0 - execute the directed fault specified in FC.

FC indicates (if F = 0) which of the four directed faults is to be executed (DF0-DF3).

2.3.6.4 The Descriptor Segment Base Register (DSBR)

The DSBR assembly shall specify the descriptor segment for a process in execution. The DSBR assembly shall contain the memory location and size of the descriptor segment, indicate if the descriptor segment is a paged segment and hold a stack segment number for the process. The format of a pair of words in memory which may be used to load the DSBR assembly shall be:



where:

DSBR.ADDR the interpretation of this field depends upon the state of the DSBR.U bit.

If the descriptor segment is unpaged (DSBR.U=1), this field specifies the base address of the descriptor segment (bit 23 is ignored).

If the descriptor segment is paged (DSBR.U=0), $DSBR.ADDR_{00-23}$ shall specify the base address of the descriptor segment page table (bit 23 is ignored).

$DSBR.BND_{00-13}$ this register shall specify the highest 0 mod 16 descriptor segment address which can be accessed without causing an out of bounds fault.

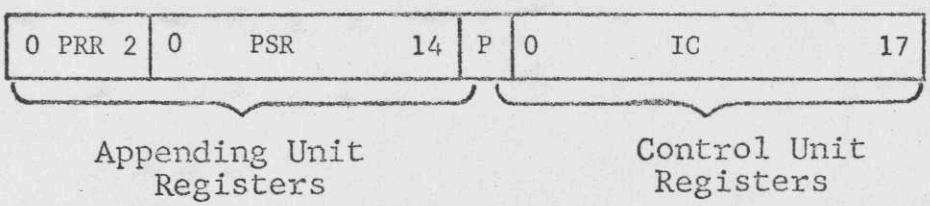
DSBR.U this register (1 bit) shall specify whether or not the descriptor segment for a process is unpaged (DSBR.U=1) or paged (DSBR.U=0).

The contents of the DSBR assembly shall be affected by the LDBR instruction only.

DSBR.STACK this register shall specify as the upper 12 bits of the 15-bit stack segment number. This register is used only during the execution of the call instruction (Section 2.5.2).

2.3.6.5 The Procedure Pointer Register (PPR)

The PPR shall be a concatenation of registers from the control unit and the appending unit which describe the current procedure segment. The PPR shall have the following format:



PRR_{00-02} Procedure Ring Register - The appending unit register shall specify the ring (level of privilege) in which the current procedure is executing. The contents of PRR may be increased (lower level of privilege) by the RTCD and RCU instructions only (Section 2.5). PRR may be set to a lower value (greater privilege) by the call instruction and PRR shall be forced to zero with the recognition of a fault or interrupt.

PSR_{00-14} Procedure Segment Register - This appending unit register specifies the segment number of current procedure segment. PSR is used as the effective segment number (ESN) for all references to the procedure segment. The PSR register contents shall be altered by:

a. A transfer of control (TRA, RET, CALL, TMI, TSX,etc.) instruction which is executed in append mode or causes the processor to enter the append mode via bit 29=1 or an ITS modifier.

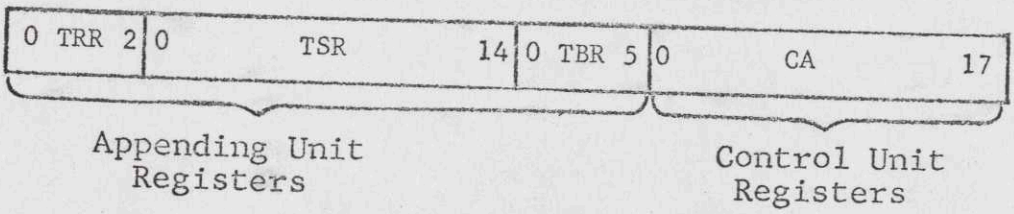
b. The execution of the RCU or RTCD instructions.

ICO_{00-17} This control unit register shall hold the relative address of the current procedure instruction and shall be altered as described in the 6000 EIS Processor EPS-1.

P This appending unit register specifies if the current procedure segment is privileged; that is, executing in ring zero with SDW.P=1. P is set or reset any time the contents of PSR or PRR is altered by a transfer, RTCD, RCU or CALL instruction.

2.3.6.6 The Temporary Pointer Register (TPR)

The TPR assembly shall be a concatenation of appending unit and control unit registers which describe each segment reference by the processor. The format of the TPR shall be:



where:

TRR_{00-02} The Temporary Ring Register - This appending unit register shall hold the highest effective ring number (ERN) encountered during an operand or instruction or indirect word address preparation sequence.

TSR_{00-14} Temporary Segment Register - This appending unit register shall specify the effective segment number (ESN) generated for each segment reference.

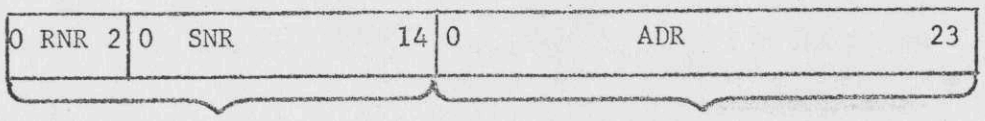
TBR_{00-05} Temporary Bit Register - This appending unit register shall specify the effective bit number generated during an operand address preparation sequence. This register shall reflect the value of the bit number specified by the last ITS modifier or Pointer Register (PR_n) referenced during an instruction execution sequence or by an ITP modifier.

CA_{00-17} Computed Address - CA is the address generated by the control unit which specifies a relative address within the segment accessed. (See 2.3.9.1.)

At the completion of an address preparation sequence for an instruction or operand (indirection completed) TPR holds the ERN, EBN, ESN and EA values to be used by the appending unit instructions. (See 2.3.9, 2.5 and 3.4.)

2.3.6.7 The Eight Pointer Registers(PRn)

The PRn assembly shall be a group of eight registers each of which is a concatenation of appending and control unit registers. These program accessible registers shall provide the information which will enable a procedure segment to directly access eight external segments through the use of bits 29 and 00-03 of an instruction (Section 2.3.3.2). The format of each pointer register shall be:



where:

RNR Ring Number Register - These eight appending unit registers specify the maximum level of privilege (smallest ring number) which can be assigned to a procedure segment attempting to access the segment referenced by this Pointer Register. That is, if RNRn is

greater (less privileged) than the Procedure Ring Register (PRR) when bits 00-02 and 29 reference PRn, then the effective ring number (ERN) for the segment access is RNRn.

The contents of the RNR's shall be altered only as specified by the following instructions which are described in Section 3.4: LPRPn, EPBPn, EPPn, RTCD, LPRI and TSPn.

SNR — Segment Number Registers - Each of these eight appending unit registers specifies the segment number of the segment referenced by that pointer register. The contents of the eight SNR's shall be altered only as specified by the following instructions which are described in Section 3.4: EASPn, EPBPn, LPRPn, LPRI and TSPn.

ADR — Address Registers - These eight control unit registers specify a relative word (bits 00-17), character (bits 18-19) and bit (bits 20-23) offsets to be used in reference to the external segments. The ADRn performs the function supplied by the internal base register in the 645 Processor.

The contents of the eight ADR registers shall be changed only as follows:

a. Execution of the instructions specified in the 6000 EIS Processor EPS-1 which affect these registers.

b. The ADR's shall be altered as specified in the following instructions described in Section 3.4.

ADWPn, EAWPn, EPPn, EPBPn, LPRPn, LPRI and TSPn.

2.3.6.8 The SDW Associative Memory Assemblies (SDWAM)

To reduce the time required by the appending unit to generate a required memory address for a PTW or a reference to a nonpaged object segment, the 645F Processor shall provide a content addressable memory to retain up to 16 SDW's on a most recent reference basis. Each of the 16 SDW register assemblies shall contain:

a. The necessary address and control fields for generating the proper address and access control information for a reference to the applicable segment or segment page table.

b. A content addressable pointer field which shall contain the segment number for the segment reference.

c. A 4-bit usage count field which indicates the relative usage of this register in relation to the other 15 registers. All 16 registers shall contain at all times a unique count (0000 through 1111) where the register with the count of 1111 shall indicate the one most recently referenced or loaded and the count of 0000 shall indicate the register least most recently referenced.

d. A full bit which indicates when the information in the register is valid.

The SDWAM shall function in the following manner:

a. Each time the processor references a SDW in memory it shall place this SDW in the least most recently used SDWAM register (usage = 0000) and set its usage count to indicate that it was the last SDW referenced (usage = 1111). In addition, the other 15 AM registers shall have their usage counts decremented by one count causing a different register to have a usage count of 0000 to indicate that it will be the register to hold the next SDW fetched from main memory. In addition the effective segment number generated for the SDW fetch from main memory shall be placed in the content

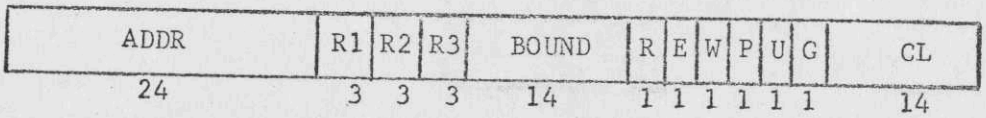
addressable pointer field of the SDWAM register and the full bit shall be set to the one state.

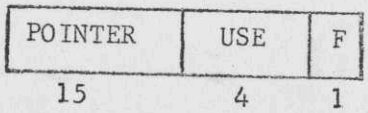
b. Each time there is a need by the appending unit address preparation logic for an SDW, the effective segment number generated by the appending unit will be compared to each of the content addressable pointer fields. If the effective segment number is identical (SDW match) to one of the 16 pointer fields and the full bit is set for that register, the SDWAM contents will be read out to the address preparation logic, the usage count for that register will be set = 1111, and the usage counts of all other registers whose value is greater than the matched register's original value will be decremented by one.

c. Each time the processor executes the LDBR or CAMS instruction all 16 full bits shall be set to the zero state and the 16 usage counts shall be set to 16 unique and predetermined values.

The contents of the SDW register assemblies shall be altered as specified by the LSDR and LSDP instructions in Section 3.4.

The format of each of the 16 SDWAM registers shall be:





where:

ADDR, R1, R2, R3, BOUND, R,E,W,P,U,G and CL are the contents of the corresponding fields of an SDW fetched from main memory (see 2.3.6.2).

POINTER: is the effective segment number generated when this SDW was fetched from main memory.

USE: is the relative usage of this segment in relation to the other 15 SDW registers, where 15_{10} (1111) is the most recently referenced. If USE = 0000, this register shall be used for storage of the next SDW fetched from main memory.

F is the full/empty bit.

0 - this register does not contain valid information. Do not consider when checking for SDW.

1 - this register contains valid information. Consider it when checking for an SDW.

2.3.6.9 The PTW Associative Memory Assemblies (PTWAM)

To further reduce the time required by the appending unit to generate operand and instruction addresses for paged object segments, the 645F Processor shall provide a content addressable memory to retain up to 16 PTW's on the most recent reference basis.

Each of the 16 PTW registers shall contain:

a. The address field and modified bit of the PTW.

b. A content addressable field which shall contain the segment number and the page number for the page which this PTW references.

c. A 4-bit usage count identical to the SDWAM usage count.

d. A full bit which indicates when the information in the register is valid.

The PTWAM shall function as the SDWAM except for the following:

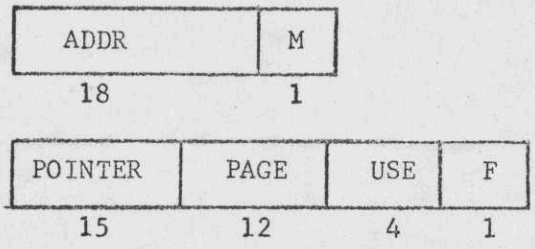
a. When a PTW is written into a PTWAM register the page number field of the computed address (see 2.3.2.4) is also placed in the content addressable field of the PTWAM register along with the effective segment number.

b. When comparing for the presence of a desired PTW both the effective segment number and the page field of the computed address (CA.PG) are compared against the PTWAM content addressable fields.

c. All 16 full bits of the PTWAM are reset by either the LDBR or the CAMP instructions.

The contents of the PTW register assemblies shall be altered as specified by the LPTR and LPTP instructions in Section 3.4.

The format of each of the 16 PTWAM registers shall be:



where:

ADDR and M are the contents of the corresponding fields PTW fetched from main memory (see 2.3.6.3).

POINTER, USE, and F have the same meaning as the associative memory SDW (see Section 2.3.6.8).

PAGE is the page number to which this PTW refers. (See Section 2.3.9.1.) Certain bits of this field shall be forced to zero depending upon the designated page size as shown in Table 2.3.6.8.

Page Size	Bits of Page Forced to Zero
64	None
128	11
256	10-11
512	09-11
1024	08-11
2048	07-11
4096	06-11

2.3.7 Absolute Address Preparation Mode

The Absolute Address Mode is in effect whenever the Absolute Indicator is set to a one or the control unit is in the Temporary Absolute Mode to service faults or interrupts.

In the Absolute Address Preparation Mode, the appending unit shall not be active in the address preparation sequence and shall not effect the generation of memory zones, addresses or commands. This mode shall be the only mode when the maintenance panel mode switch is set to the GECOS position. However, when the mode switch is set to the MULTICS position the Absolute Address Mode shall be used when certain requirements are met as described in Section 2.2.

In this mode the computed address generated by the control unit shall be the address used to access main memory (see 2.3.9.1).

Finally, when the Absolute Address Mode and the MULTICS operational mode is in effect, the instructions which effect the appending unit registers shall perform as specified in Section 2.5.

2.3.8 Append Address Preparation Mode

The Append Address Mode is in effect whenever the Absolute Indicator is set to the zero state and the control unit is not in the Temporary Absolute Mode to service a fault or interrupt.

In the Append Address Preparation Mode, all addresses shall be segmented and ringed, the appending unit shall control the memory zone and command lines for all appending word accesses (DSPTW, PTW, SPW) and the appending unit shall control the memory address lines for all memory accesses.

2.3.9 The Address Preparation Sequence

The address preparation scheme in the 645F Processor can be thought of as two separate phases where the object of one phase is the generation by the control unit of the relative address within the object segment (computed address) and the other phase consists of utilizing the appending unit to generate the absolute base address of either the nonpaged object segment or the object page.

2.3.9.1 Computed Address Generation

The computed address (CA) in the 645F Processor is the 18-bit address generated by control unit for an instruction, indirect word or operand fetch from (or store to) memory. The computed address is the absolute memory address when the processor is in either the GECOS mode or in the MULTICS mode with the Absolute/Append Indicator set to the Absolute state. When the processor is in the MULTICS mode and the Absolute/Append indicator is set to the Append state, the computed address is a relative address within a segment and is interpreted in one of two ways:

1. Unpaged Segment -- For unpaged segments CA specifies an 18-bit address relative to the unpaged segment base address.

2. Paged Segment -- For paged segments CA is divided into a page number field (CA.PG) and a word number field (CA.WD). The CA.PG field specifies the address of a PTW relative to the base address of the page table and the CA.WD field specifies the address of the required instruction, operand or indirect word relative to the base address of the object page.

The length of the CA.PG and CA.WD fields for paged segments is determined by page size currently hardwired into the processor as shown in Table 2.3.9.1. The initial page size for the 645F Processor shall be 1024 word pages, however, this size may be altered by a Field Change Order.

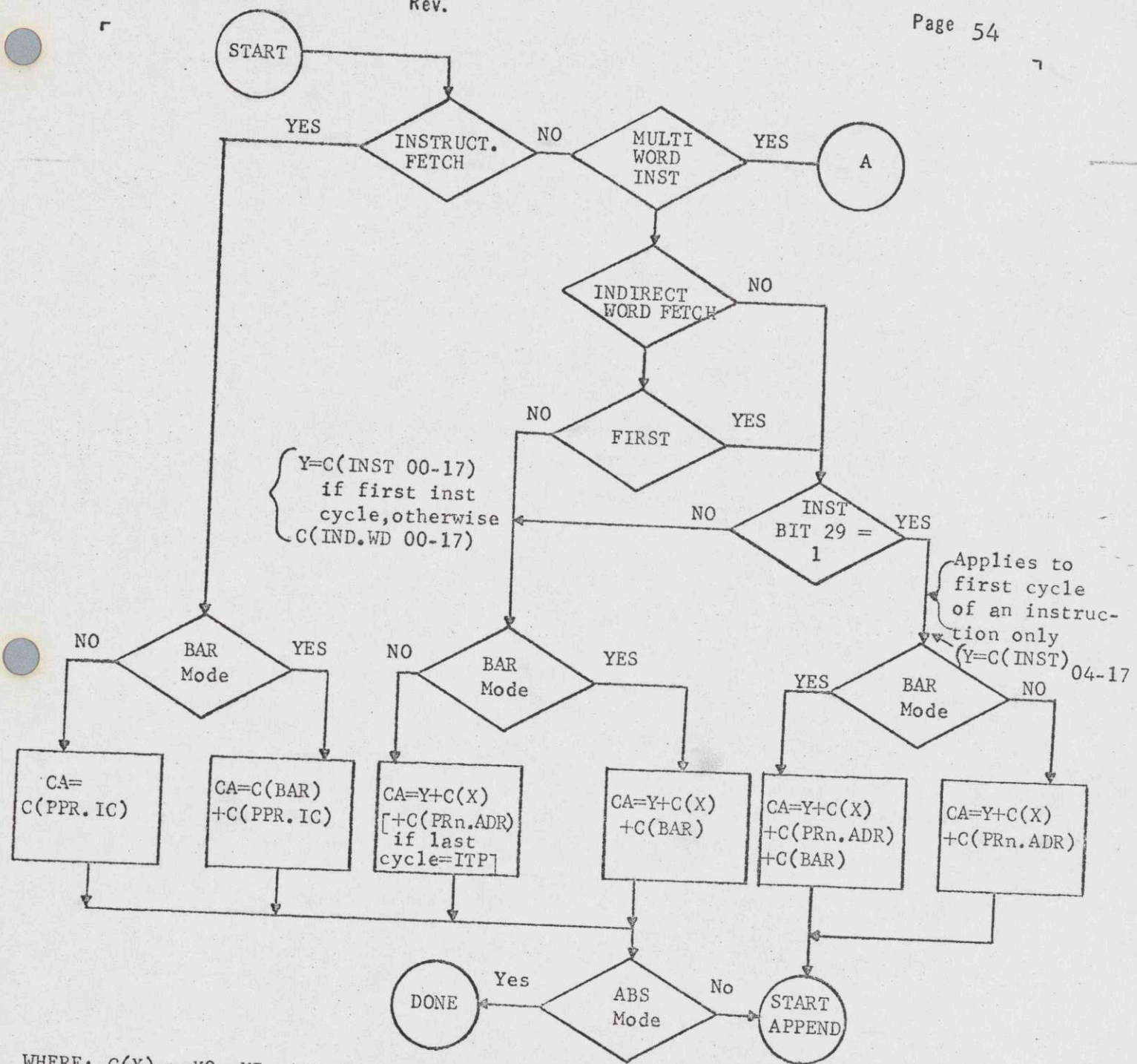
Table 2.3.9.1

Paged Mode Interpreation of the Computed Address

Page Size (Words)	CA Page Field	CA Word Field
64	CA00-11	CA12-17
128	CA00-10	CA11-17
256	CA00-09	CA10-17
512	CA00-08	CA09-17
1024	CA00-07	CA08-17
2048	CA00-06	CA07-17
4096	CA00-05	CA06-17

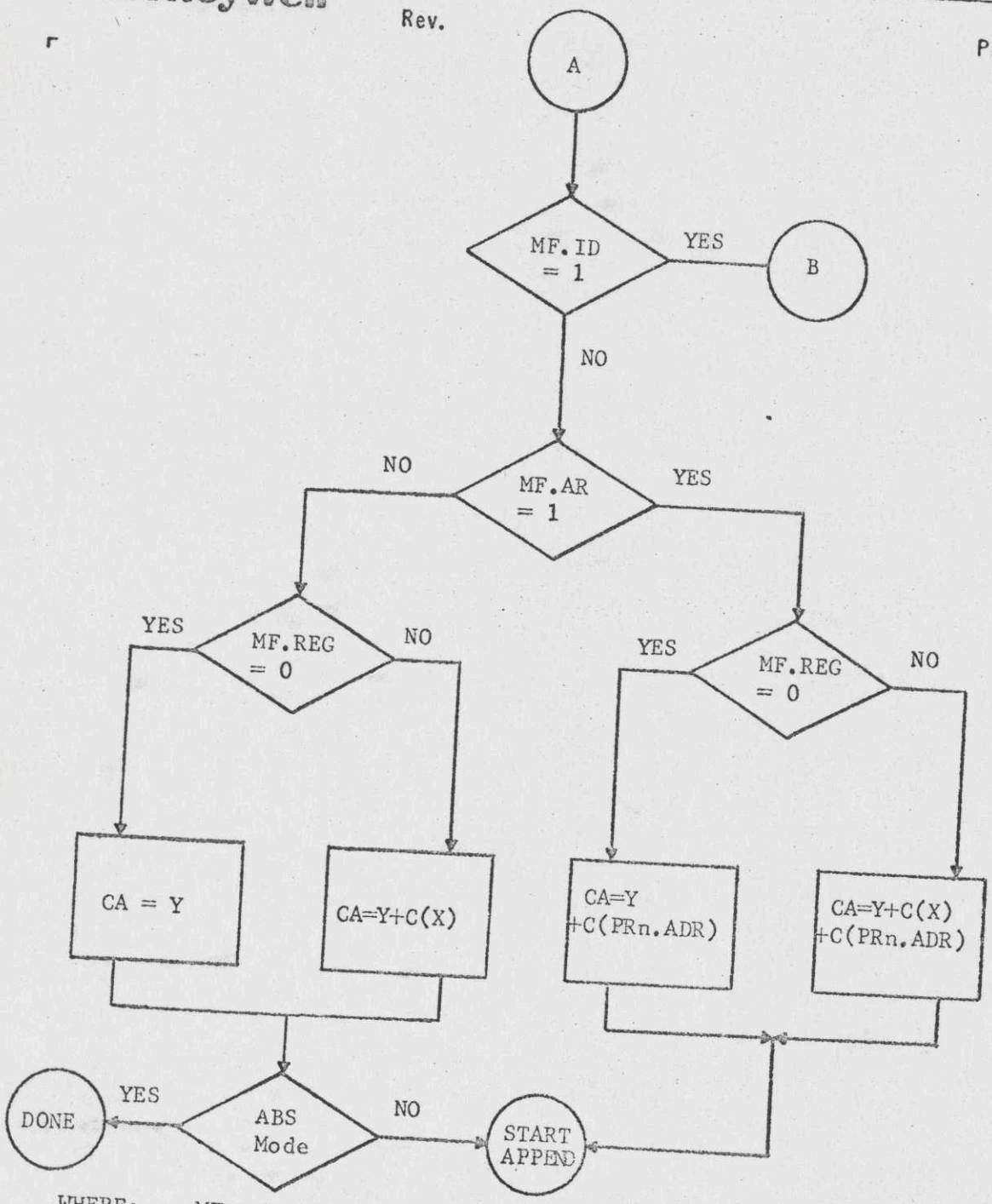
A complete description of the generation of the computed address can be found in the EPS 1 for the 6000 EIS Processor (43A232719) and the EPS 1 for the 655 Processor (43A219601).

CA for instruction fetches shall be derived from the instruction counter (PPR.IC) when the BAR Mode indicator is on and shall be derived from the addition of PPR.IC and base address register (BAR) when the BAR Mode indicator is off.



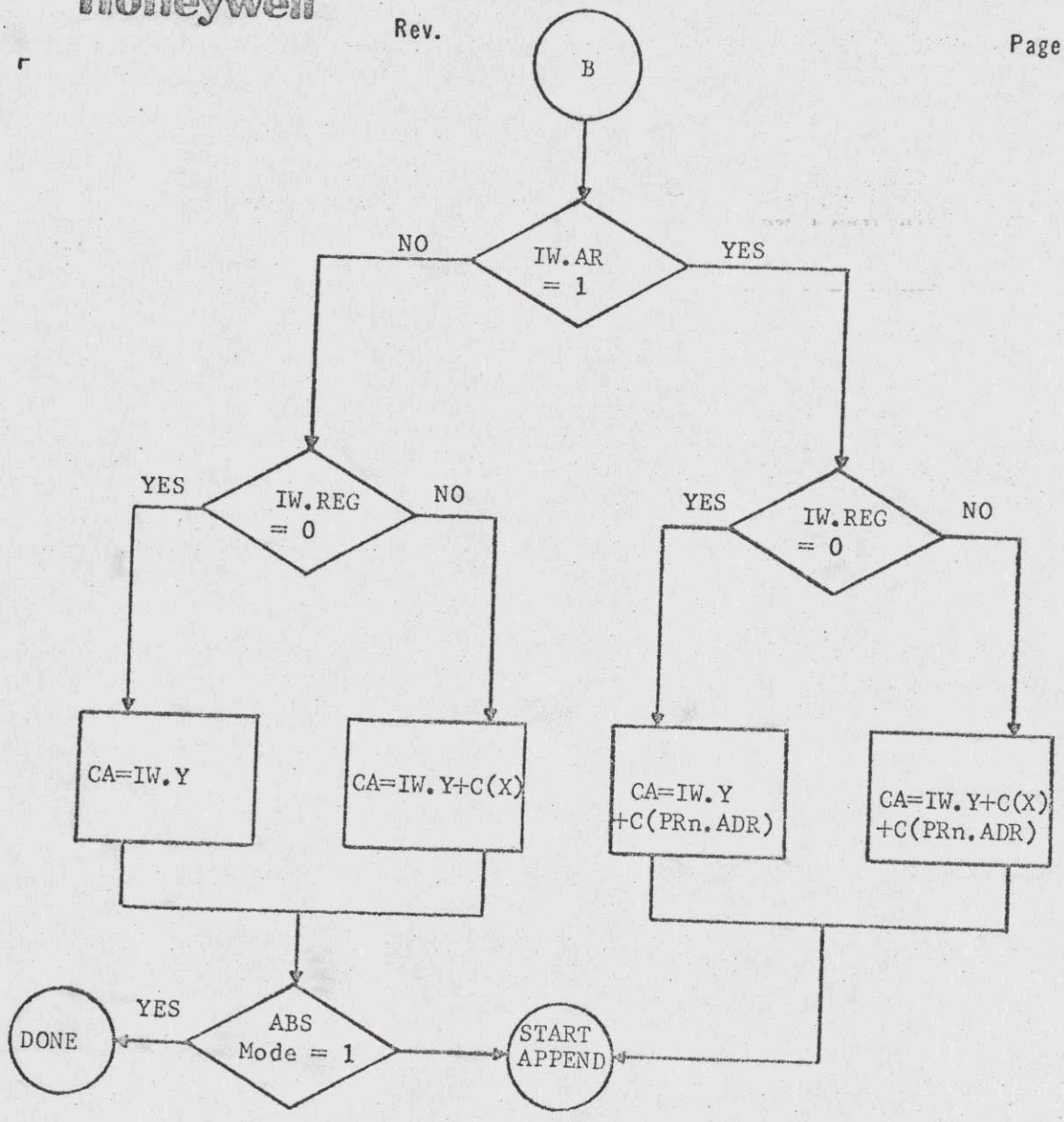
WHERE: C(X) = X0, X7, AU, QL, etc. specified by R, RI, IR or IT modification.
PRn.ADR = the address register portion of pointer register n +C(PRn.ADR) is specified by bits 00-02 of the instruction where Y03 is extended to Y00, Y01, Y02. BAR is the Base Address Register in the CU.

Figure 2.3.9.1a - Computed Address Generation for Single-Word Instructions and Their Operands



WHERE: MF = Modification Field
Y = Y Field of Operand Descriptor
C(X) = Register specified by Reg Field in MFI
+C(PRn.ADR) is specified by bits 00-02 of the operand descriptor where Y03 is extended to Y00, Y01, Y02.

Figure 2.3.9.1b - Computed Address Generation for Operand of Multiword Instruction



WHERE: IW = Indirect Word specifying an operand descriptor
IW.Y = Y Field of IW
C(X) = Register specified by Reg Field IW
+C(PRn.ADR) is specified by bits 00-02 of IW, where Y03 is extended to Y00, Y01, Y02.

Figure 2.3.9.1c - Computed Address Generation for Operand Descriptor of Multiword Instruction

CA for operand and indirect word is derived from an instruction or indirect word address fetch (y) and a specified combination of an index modifier (X0, X7, AU, Q, etc.) an address register (PRn.ADR) and the BAR (in BAR mode). Figure 2.3.9.1 shows the general flow for the generation of the CA for an instruction, operand or indirect word.

2.3.9.2 Append Address Preparation Sequences

Once the computed address is generated, the process of generating the final address for an instruction, operand or indirect word can involve up to five preliminary memory cycles without the use of the Associative Memory. When the Associative Memory contains the necessary Segment Descriptor Word (SDW) or Page Table Word (PTW) the number of preliminary cycles are reduced or eliminated.

The possible appending sequence combinations are illustrated by the flowchart in Figure 2.3.9.2a. This chart assumes that directed faults (page not in core), access violation faults (illegal ring crossing or prohibited segment use), out of bound faults (relative segment address is too high or the address specified is not configured to the system) or memory parity faults do not occur. Directed faults can occur following fetching a DSPTW, SDW or PTW from memory and access violation faults can occur after fetching a SDW. (See Section 2.4.)

It is in this chart that the value of the associative memory can be seen by following the various paths to FAP or FAN. When the correct SDW (or SDW-PTW pair) is found, no intermediate cycles are necessary to generate an address in the object segment or page unless the modified bit needs to be set in the PTW.

There are a total of eight different appending cycles which can initiate a memory cycle. Two of these (FAN, FAP) generate the memory address of instructions, operands and indirect words; four of these cycles (NSDW, PSDW, PTW and DSPTW) generate a memory address to fetch appending words; and two of the cycles (MDSPTW and MPTW) are used to regenerate the address of a DSPTW or PTW to set the accessed or modified bits in memory. The following is a list of basic definitions for the eight cycles.

FAN - Final Address Nonpaged Cycle - The appending unit cycle which initiates a memory cycle to fetch (store) an instruction, operand or indirect word from an unpaged object segment.

FAP - Final Address Paged Cycle - The appending unit cycle which initiates a memory cycle to fetch (store) an instruction, operand or indirect word from a page in a paged object segment.

NSDW - Nonpaged SDW Cycle - The appending unit cycle which initiates a memory cycle to fetch a SDW from a nonpaged descriptor segment.

PSDW - Paged SDW Fetch Cycle - The appending unit cycle which initiates a memory cycle to fetch a SDW from a page in a paged descriptor segment.

PTW - Object Segment Page Table Word Fetch Cycle - The appending unit cycle which initiates a memory cycle to fetch a PTW from an object segment page table.

DSPTW- Descriptor Segment Page Table Word Fetch Cycle - The appending unit cycle which initiates a memory cycle to fetch a PTW from a descriptor segment page table.

MDSPTW- Modify DSPTW Cycle - The appending unit cycle which is entered following the first access of a DSPTW in memory. This cycle sets the accessed bit (DSPTW.A) which the operating system can later use to verify that main memory space taken up by this portion of the descriptor segment has in fact been used.

MPTW - Modify PTW Cycle - The appending unit cycle which is entered following the first access of a PTW in memory or in the first alteration (store) of the page of object segment specified by the PTW. This cycle sets the accessed bit (PTW.A) and the modified bit (PTW.M) as dictated by the type of operation being performed with the page (read or write) so that the operating system can later verify that this page has been recently used and whether or not the page still exists unaltered in memory.

Figures 2.3.9.2a through 2.3.9.2i show the detailed formation of addresses for the various appending word and final address cycles. The MDSPTW and MPTW cycles are not shown because they are identical to the DSPTW and PTW cycles.

It can be seen that a segment boundary check is made on all cycles except the PSDW cycle even though the same boundary check is sometimes made on two successive cycles - PTW and FAP. The reason for this is that the FAP cycle can be entered without going through the PTW cycle if the PTW resides in the associative memory.

Effective Segment Number Generation

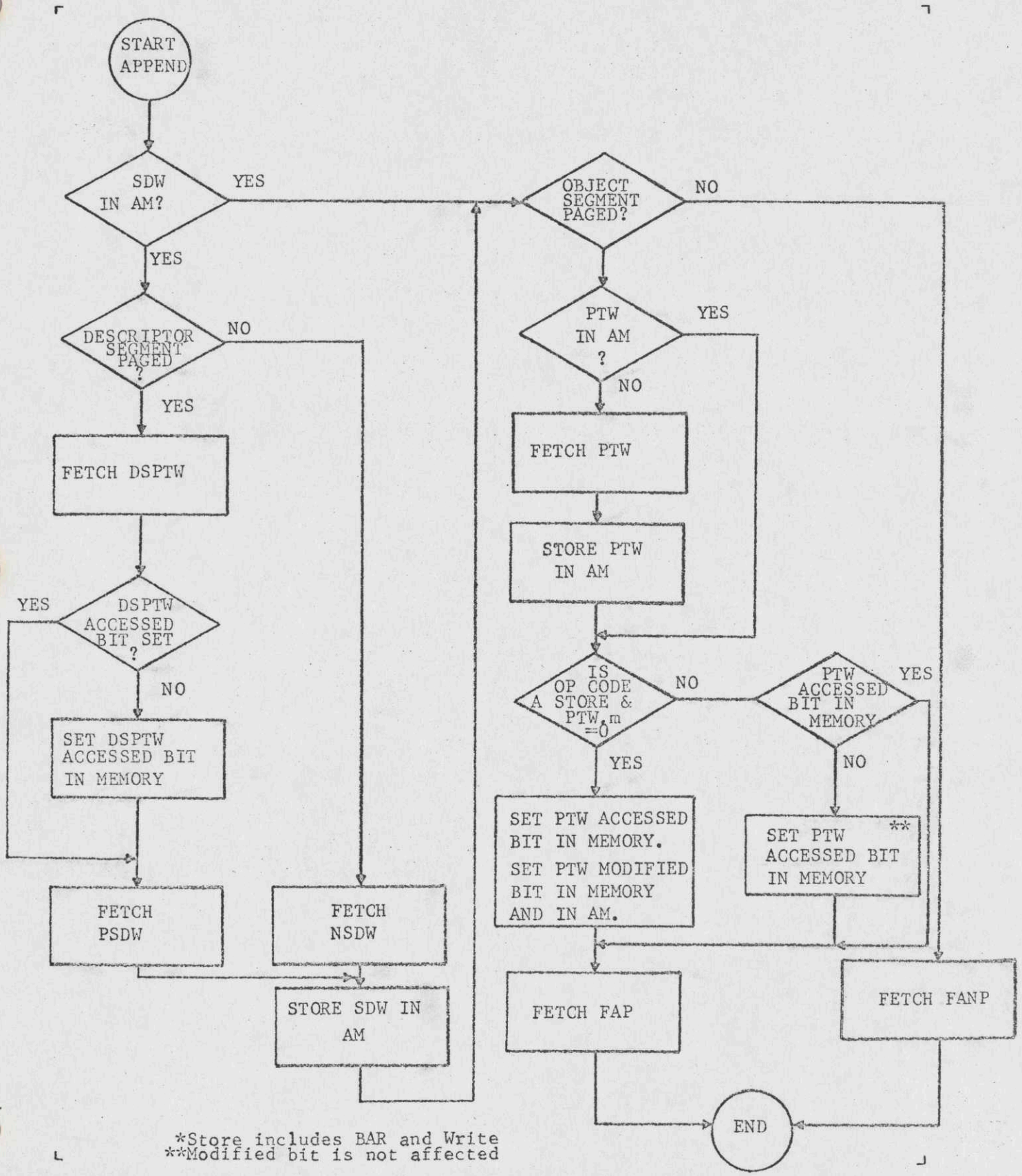
The ESN for a given descriptor segment can be either the PSR, the TSR or one of the eight SNR's.

a. ESN from PSR - The PSR contains the effective segment number for all sequential instruction fetches. The PSR is also used for operand and indirect word fetches or stores when instruction bit 29 equals a zero and no indirect words have been accessed.

b. ESN from TSR - The TSR contains the effective segment number following the fetch of an indirect word.

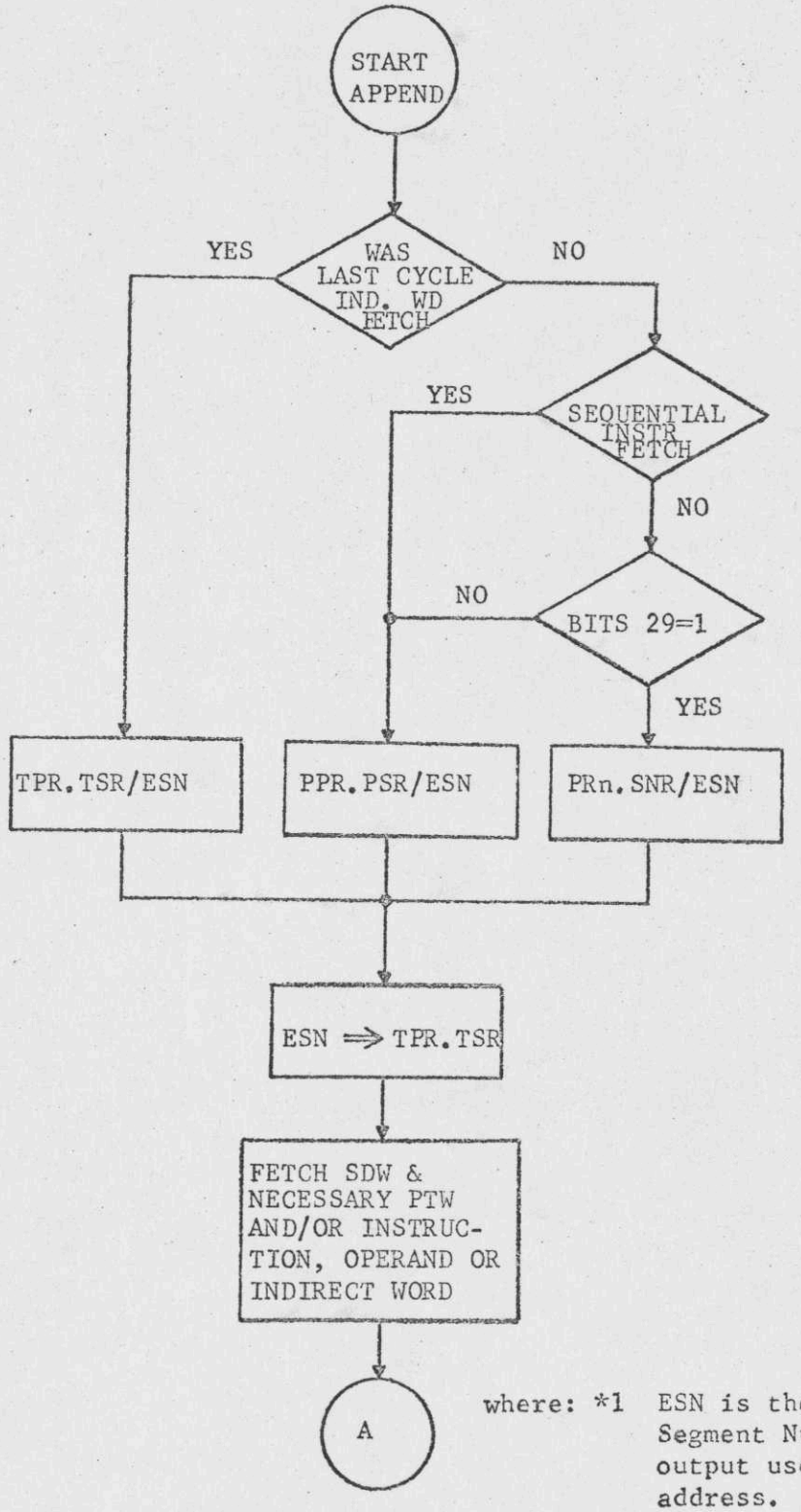
c. ESN from SNR_n - A SNR contains the effective segment number for the operands and indirect words to be fetched (or stored into) whenever bit 29 of the instruction currently being executed is a one and no indirect words have been accessed.

d. ESN Generation following an Indirect Word Fetch - If an instruction fetches an indirect word (IR or RI), the ESN for successive cycles be the TSR. The contents of TSR will depend on the contents of the indirect word.



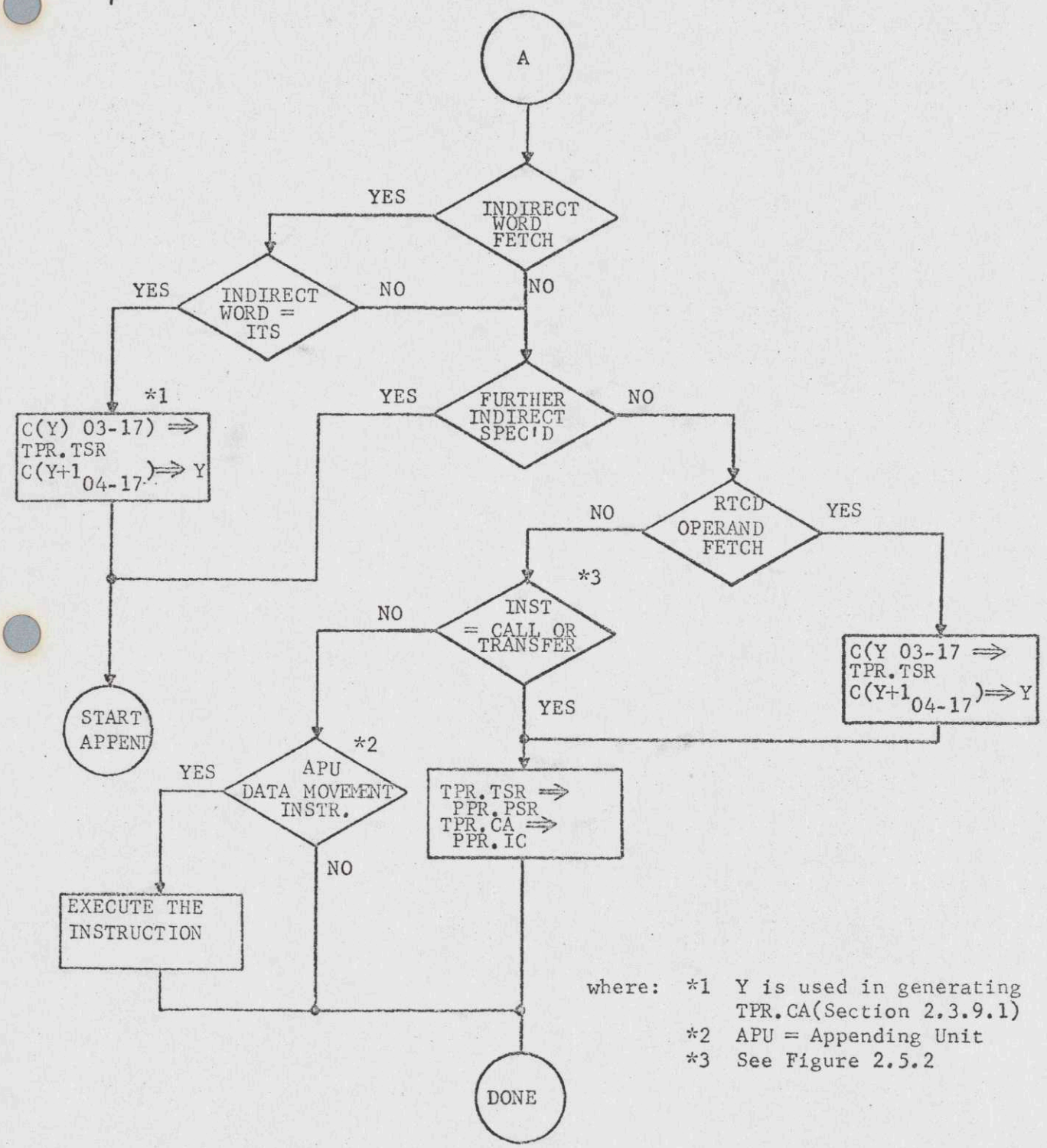
*Store includes BAR and Write
**Modified bit is not affected

Figure 2.3.9.2a - Appending Cycle Flowchart



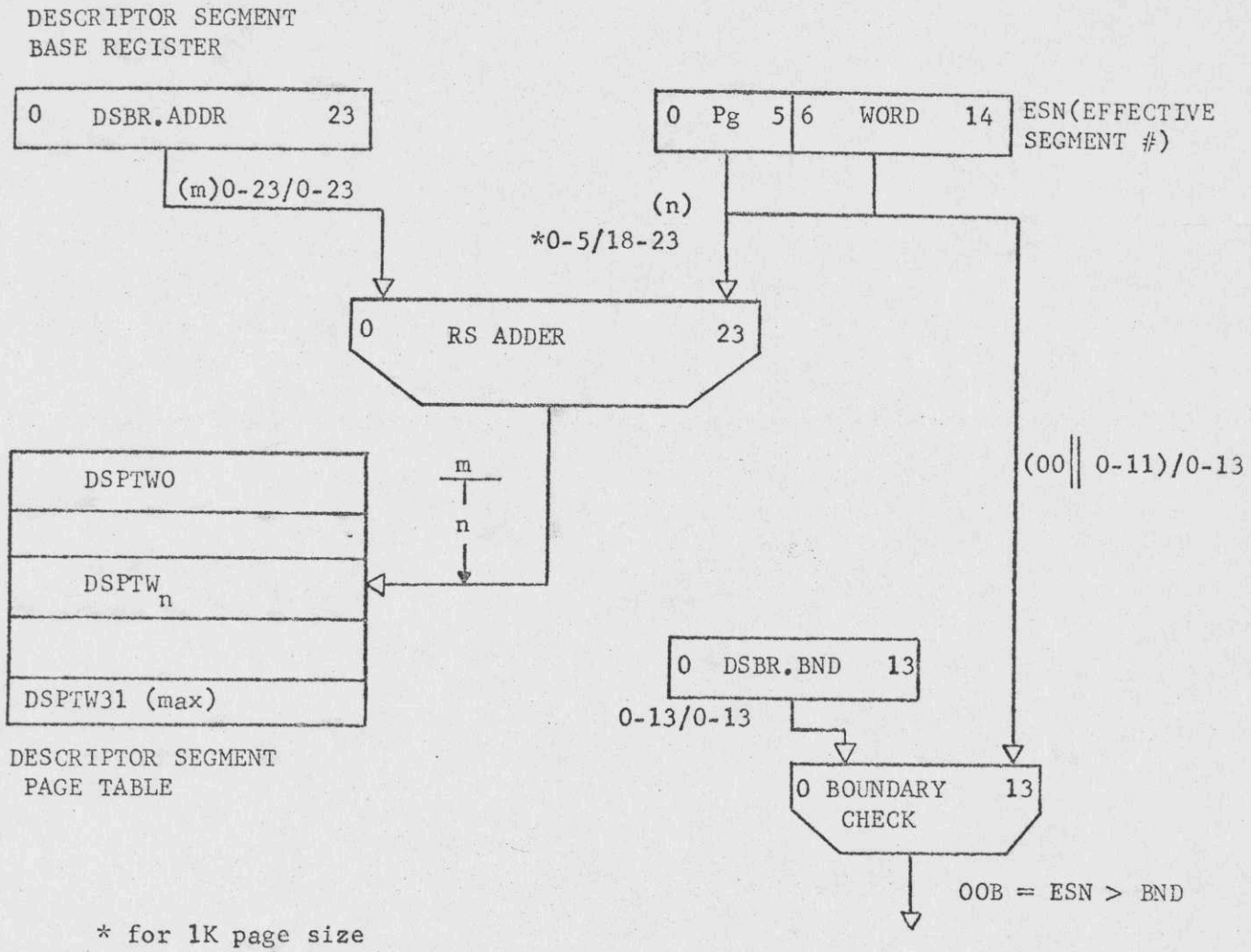
where: *1 ESN is the Effective Segment Number Switch output used as the SDW address.

Figure 2.3.9.2b - ESN Generation



where: *1 Y is used in generating TPR.CA(Section 2.3.9.1)
*2 APU = Appending Unit
*3 See Figure 2.5.2

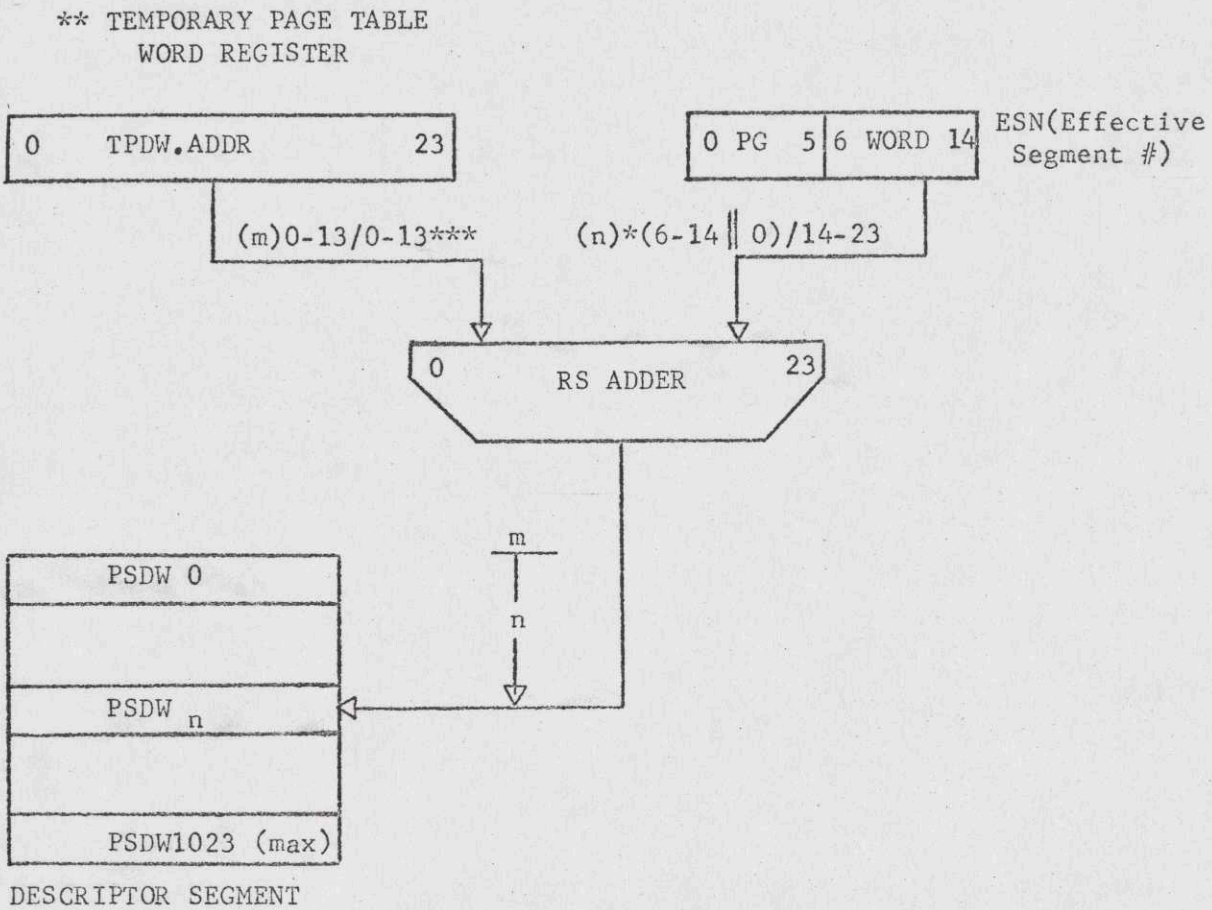
Figure 2.3.9.2c - Modification of Registers used to Generate ESN



* for 1K page size

Previous Cycle - None
Next Cycle - PSDW Fetch, MDSPTW

Figure 2.3.9.2d - Descriptor Segment Page Table Word Fetch (DSPTW)



* For 1K page size

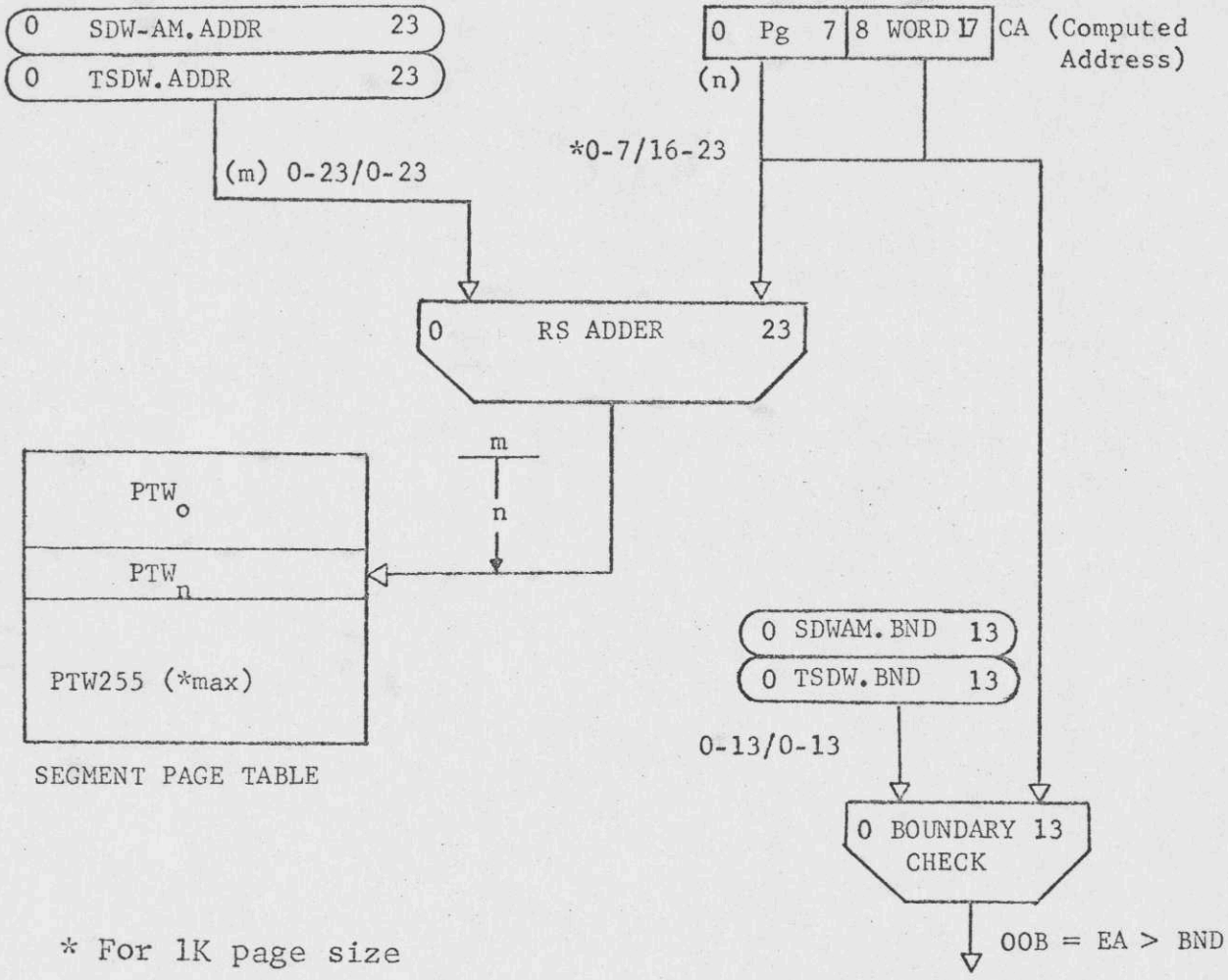
** TPTW (Temporary PTW Register) holds a DSPTW which points to the base of the Descriptor Segment (mod 1K)

*** Bits 14-23 of the DSPTW are forced to zero for 1K pages

NOTE: The boundary check done before the DSPTW fetch checked the maximum pair of SDWPS addresses

Previous Cycle - DSPTW or MDSPTW
Next Cycle - PTW or FANP or MPTW

Figure 2.3.9.2e - Paged SDW Fetch (PSDW)



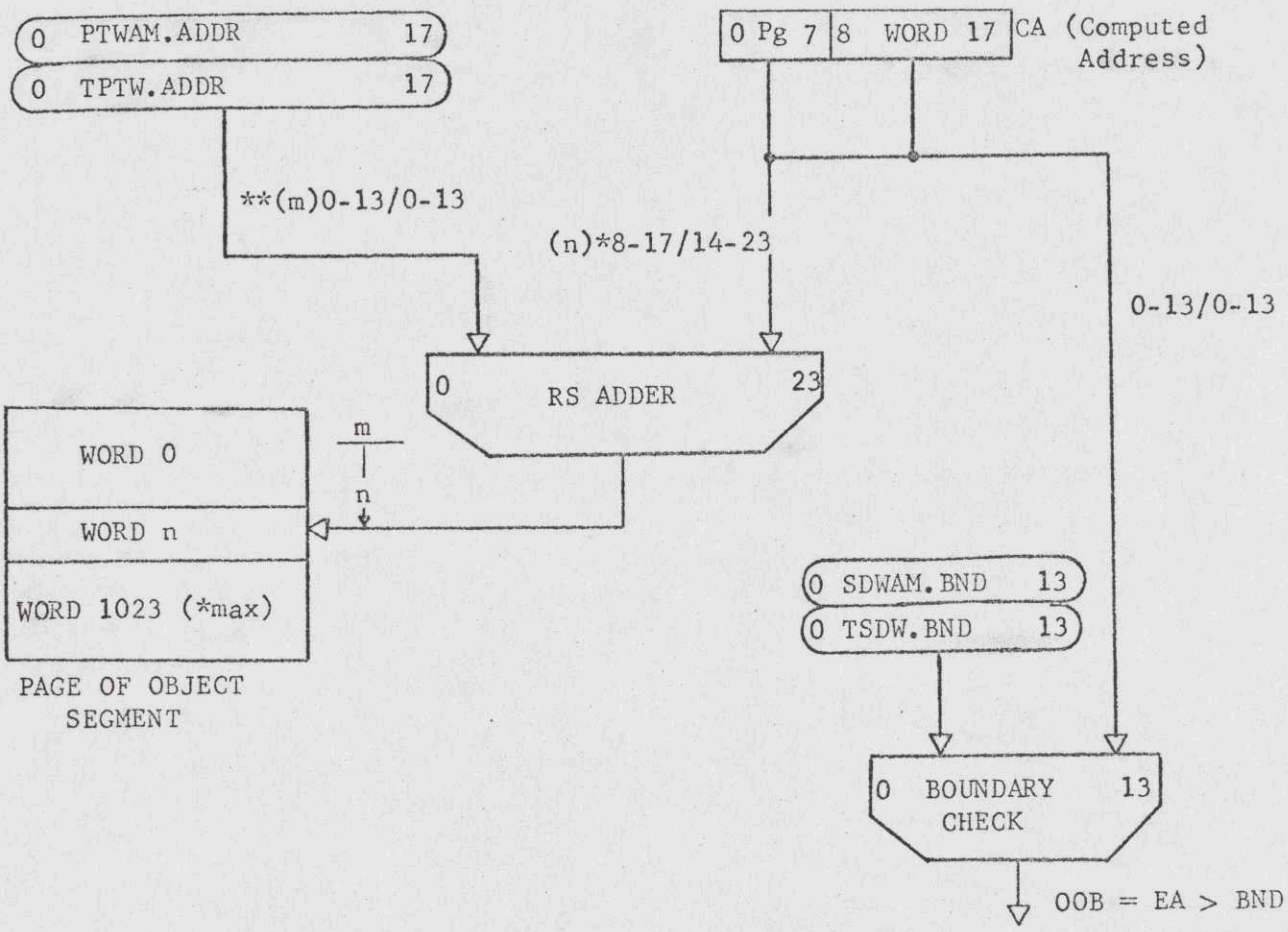
* For 1K page size

SDW-AM is used if a SDWP match occurred.

TSDW (temporary SDW Register) would hold the SDWP if it was fetched from store.

Previous cycle - NSDW or PSDW or none
Next cycle - FAP or MPTW

Figure 2.3.9.2f - Page Table Word Fetch (PTW)



* For 1K page size
** Bits 14-17 of the PTW are forced to zero for 1K page size.

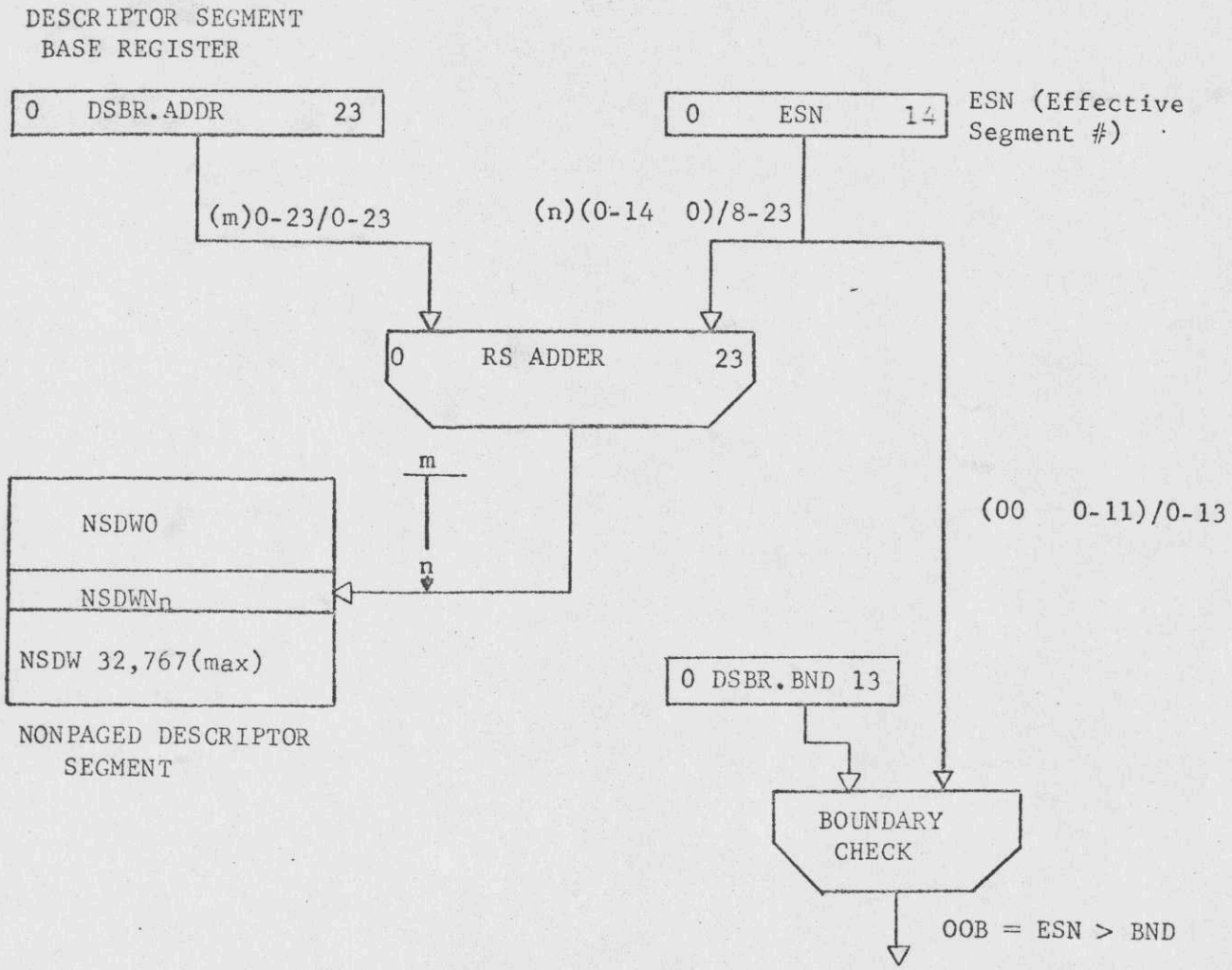
(a) use PTW-AM.ADDR and SDW.BND if PTW match

(b) use TPTW.ADDR and TPTW.BND if PTW from store (TPTW - Temporary Page Table Word Register)

Previous cycle = none or PTW or MPTW

Next cycle = None

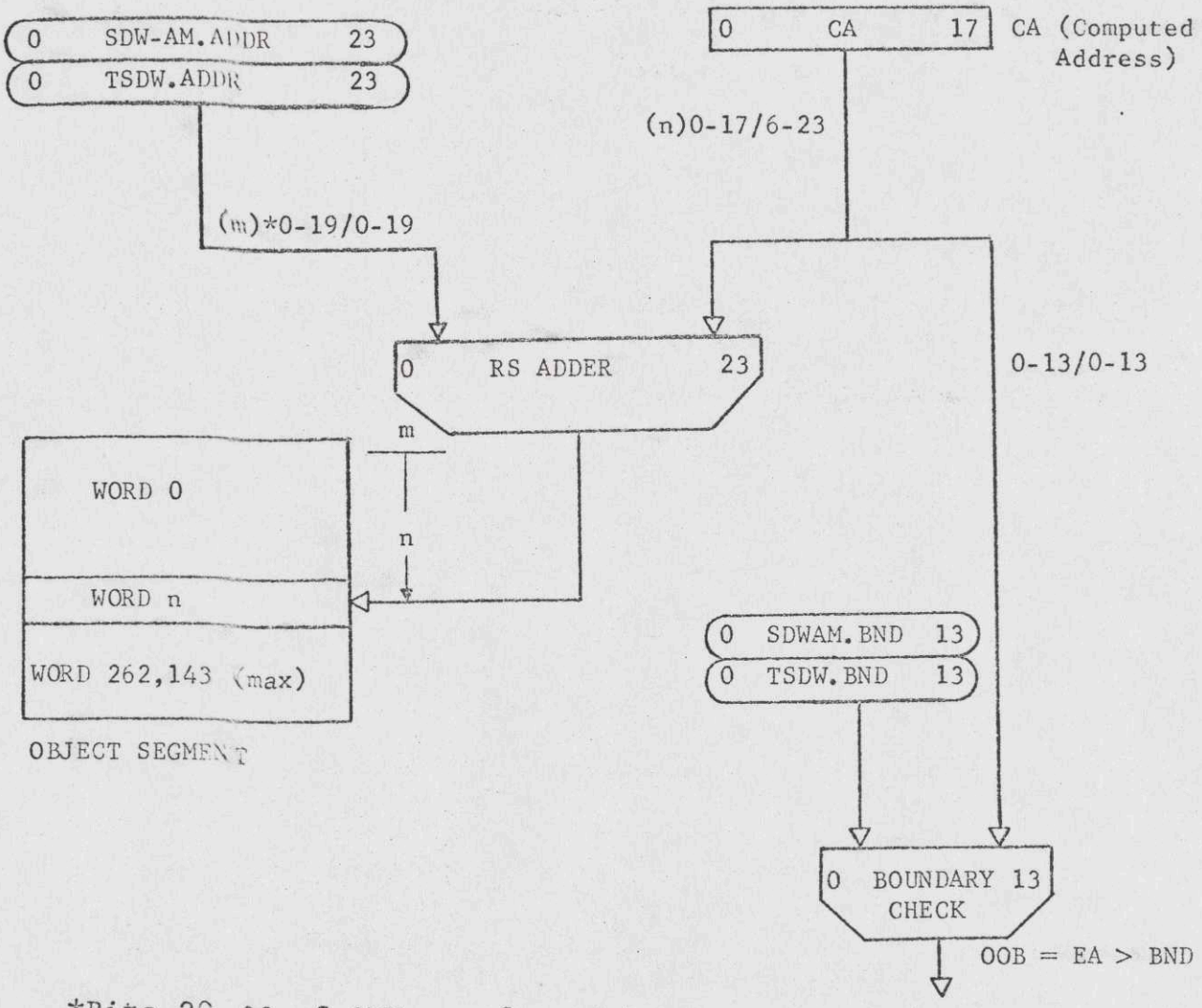
Figure 2.3.9.2g - Final Address Paged Fetch (FAP)



NOTE: The boundary check is to the maximum block of 16 NSDWS addresses.

Previous Cycle - None
Next Cycle - PTW or FANP, or MPTW

Figure 2.3.9.2h - Nonpaged SDW Fetch (NSDW)



*Bits 20-23 of SDW are forced to zero

(a) use SDW-AM if SDW match

(b) use TSDW (Temporary Segment Descriptor Word Register) if SDW from store

Previous cycle - None, PSDW, NSDW
Next cycle - None

Figure 2.3.9.2i - Nonpaged Final Address Fetch (FANP)

1. If the indirect word is not an ITS or ITP modifier TSR is loaded from ESN.

2. If the indirect word is an ITS or ITP modifier its segment number field is loaded into the TSR and used for ESN until another ITP or ITS modifier is encountered or the operand is fetched.

3. If the indirect word is an ITP modifier, ITP.PRNUM specifies the PRn.SNR which is loaded into the TSR and used for ESN until another ITP or ITS modifier is encountered or the operand is fetched.

2.4 APPENDING UNIT ACCESS CONTROL

2.4.1 Segment Boundary Control

Prior to accessing any segment, the 645F Processor shall verify that the intended access is to a location within the segment which is less than the upper limit specified for the segment. If the intended access is greater than the specified limit, the segment will not be accessed and an out-of-bounds fault shall be initiated. Boundary verification shall be performed as follows:

a. $DSBR.BND_{00-13}$ shall be compared to the most significant 12 (Figure 2.3.9.2h) bits of effective segment number (ESN_{00-13}) before accessing a DSPTW for a paged descriptor segment or a SDW for a nonpaged descriptor segment. (Out-of-bounds = ESN > BND.)

b. After a SDW is obtained from main memory or the associative memory, the boundary field of that SDW ($SDW.BND_{00-13}$) shall be compared to the most significant 14 bits of the 18 bit computed address (CA_{00-13}) before accessing any PTW for that segment or any location within the segment itself. (OOB = CA > BND)

c. If a CALL instruction is being executed and any specified indirect addressing is completed, the processor shall obtain the SDW for the "called" segment and if the gate indicator in the SDW is on (SDW.G=0) the processor shall compare the segment call limiter ($SDW.CL_{00-13}$)

to the least significant 14 bits of the computed address (CA_{00-13}). (Access Violation = CA_{00-17} $0000 \geq CL_{00-13}$)

The boundary comparisons for (a) and (b) above are illustrated in the appending cycle address preparation charts in Section 2.3.9.2 and (c) above is shown in Figure 2.4.3.2e.

2.4.2 The MULTICS Ring Structure

2.4.2.1 Theory and Definition

The MULTICS concept of protection rings, ring crossing and argument validation shall be implemented as an integral part of the paging and segmentation hardware on the 645F Processor. The hardware implementation of rings is really a further modification to the paging and segmentation hardware. However, the modification is introduced separately at this point since it involves perhaps the most significant deviation from the 645 Processor specification and, as such, deserves detailed explanation. In the current version of MULTICS running on the 645 system, the ring protection mechanism is, of necessity, completely simulated by the MULTICS software. The current system maintains, in parallel, separate descriptor segments for each ring of each process. The ring crossing is simulated by a rather costly and complex fault processing mechanism which includes the copying and validation of argument pointers and the switching of descriptor segments to simulate the effect of switching rings. The cost of the current simulation amounts to approximately 10-20 percent of the useful chargeable CPU time and contributes substantially to the overall complexity of the system. In the 645F Processor, ring crossing and argument validation shall be handled directly by the hardware without costly software intervention. As a result, a call to an inner ring will require no more CPU time than a call to a procedure in the same ring.

A MULTICS process consists of procedure and data segments which are all directly addressable through the descriptor segment of that process. However, a process may only access segment when the process is running at an appropriate level of privilege. For example, all of the segments of the hard core supervisor are shared and accessible to all MULTICS processes but only when they are executing at the highest level of privilege.

The 645F Processor hardware ring structure implementation shall allow segments to be grouped into an ordered set of collections called rings in which segments requiring the highest level of privilege to reference can only be accessed from within the innermost ring of the set. Each ring shall be identified with a ring number designating the highest level of privilege allotted to procedure segments executed in that ring. The range of ring numbers shall be from zero to seven, where ring 0 has the highest level of privilege.

The access of a user to a specific segment shall be controlled by two quantities contained in the SDW for that segment; the access attributes (for example, read, execute, write) and the ring brackets. The ring brackets of a segment are specified by three integers (R1, R2 and R3) each of which must be greater than or equal to the preceding number. The first number (R1) specifies the top (highest ring number) of the read/write bracket, the second number (R2) specifies the top of the read/execute bracket and the last number (R3) specifies the top of the call bracket.

The Read/Write Bracket (Rings 0-R1)

Attempts to read or write a segment by a procedure executing in a ring within the read/write bracket shall be allowed if the appropriate (read or write) access indicators are on for the segment being referenced. Execution of a segment by a procedure shall be permitted only if the procedure is executing at the top of the read/write bracket (R1) which is also the bottom of the read/write bracket. Note that the highest ring from which a segment can be written is specified by R1. As a result, the data in the segment is no more reliable than the procedure segments which operate in that ring.

The Read/Execute Bracket (Ring R1-R2)

Attempts to read or execute (transfer to) a segment by a procedure executing in a ring within the read/execute bracket shall be allowed if the appropriate (read and execute) indicators are on for the segment being referenced. Writing of a segment within its read/execute bracket shall be permitted only from the ring at the bottom of the bracket (R1) which is also the top of the read/write bracket. If R2 is equal to R1 the read/execute bracket shall specify a single ring (R1).

The Call Bracket (Rings(R2 + 1) - R3)

Attempts to call a (procedure) segment from another segment executing in a ring above and read/execute bracket, but within the call bracket of the procedure to be called, shall be allowed if the execute indicator of the procedure to be called is on and if the CALL instruction is used. When the CALL instruction is directed to a procedure in an inner ring which has the appropriate execute access and call bracket, the processor will automatically switch to the ring specified as the R2 of the procedure being called. The call bracket and the CALL instruction are the only means (except for faults) by which control can be passed from an outer ring to an inner (more privileged) ring. If R3 is equal to R2, the call bracket is null and the procedure cannot be called from an outer ring.

Summary of Ring Brackets

Assuming that the SDW for a segment has the appropriate indicators on (read, write or execute) and that $R1 \leq R2 \leq R3$, the following list summarizes the requirements of the three ring brackets. Any attempt by a segment to violate a segment ring requirement shall be detected and an illegal procedure fault shall be initiated.

Writing is permitted from a ring within the read/write bracket only (that is, ERN not > R1).

Reading is permitted from a ring within the read/write bracket or the read/execute braket (that is, ERN not > R2).

Execution (or transfer of control) is permitted only from a ring within the read/execute bracket (that is, R1 $\leq$ ERN $\leq$ R2).

Calling (via CALL only) is permitted from a ring within the read/execute or call brackets (that is, R1 $\leq$ ERN $\leq$ R3).

The CALL instruction is the only instruction which may be used to access a segment in a ring within its call bracket (that is, R2 $\leq$ ERN $\leq$ R3).

No access is permitted to a segment from a ring higher than the call bracket (that is, ERN $>$ R3).

2.4.2.2 Ring Structure Hardware Implementation

In order to implement the hardware ring structure requirements as specified in 2.4.2.1, registers shall be supplied as specified in Section 2.3.6.5 through Section 2.3.6.7 to hold ring information for the PPR, TPR and PRn register assemblies. In addition, the necessary hardware for the detection of access violations as shown in Figures 2.4.2.2 shall be supplied in the processor appending unit.

The notes specified in Figures 2.4.2.2 are:

*1 n - specified by instruction bits 00-02

*2 from main memory or SDWAM

*3 excludes XEC, XED, RTCD and CALL

*4 includes store portion of RAR for address modification or operand

*5 PSR = TSR implies that read is from current procedure segment

*6 ERN is a switch output. XXX/ERN implies the contents of XXX is switched to the ERN output

*7 RSDWH.R1 is a 3-bit register used to remember R1 of an SDW in anticipation of an ITS, ITP or an RTCD

*8 RIB is a 3-bit register used to remember the ring number C(Y_{18-20}) from an its word or an RTCD operand

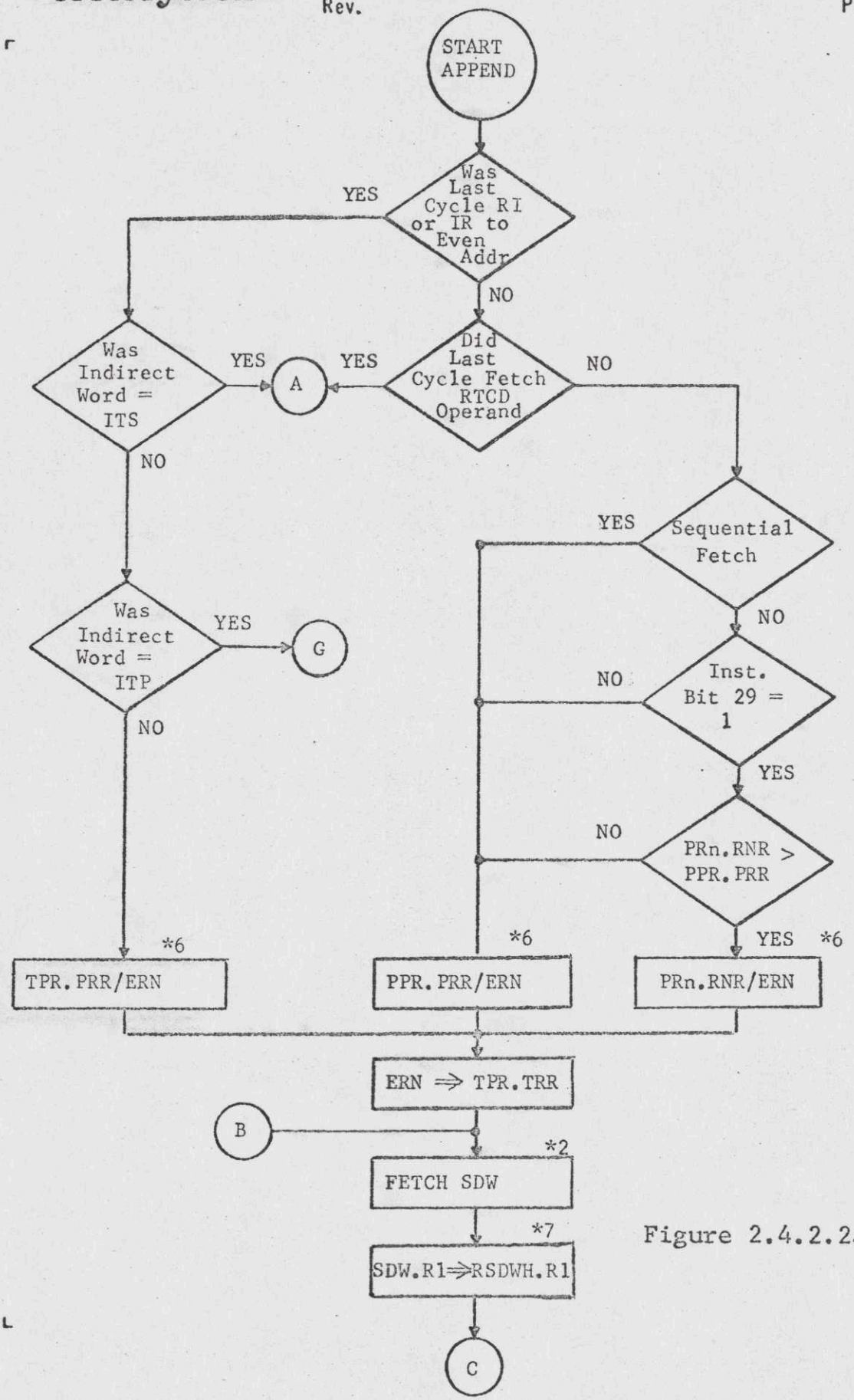


Figure 2.4.2.2a

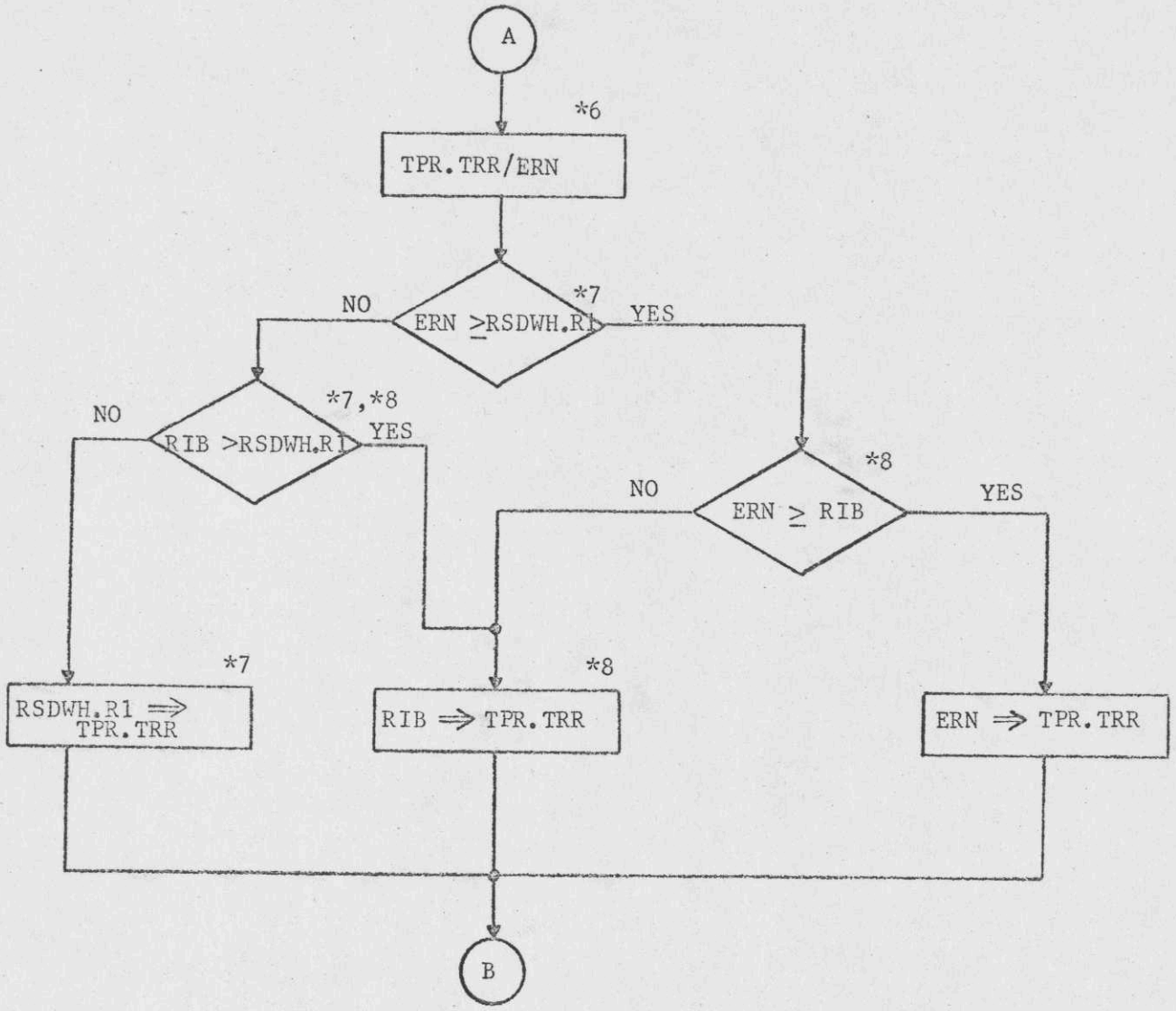


Figure 2.4.2.2b

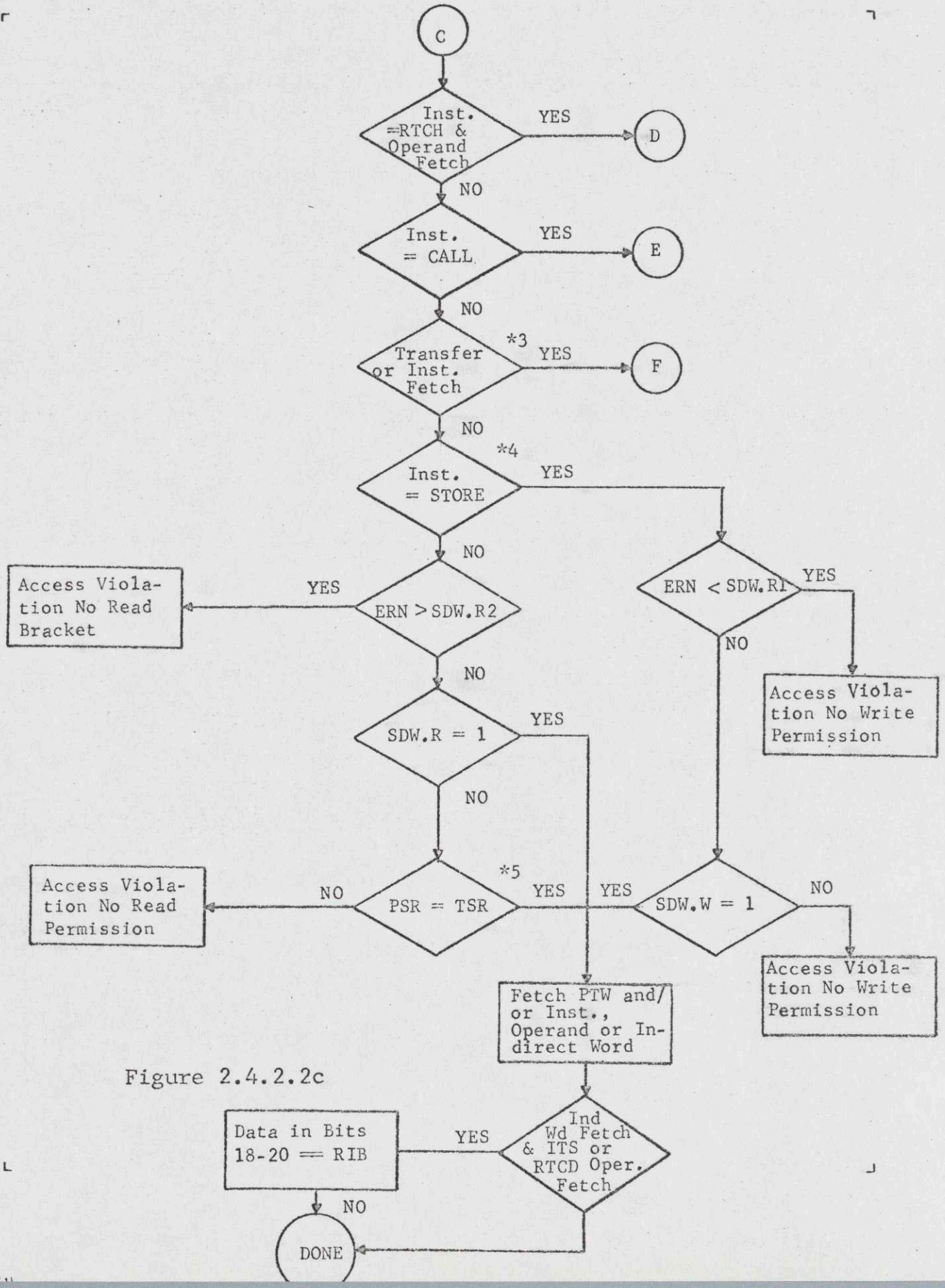


Figure 2.4.2.2c

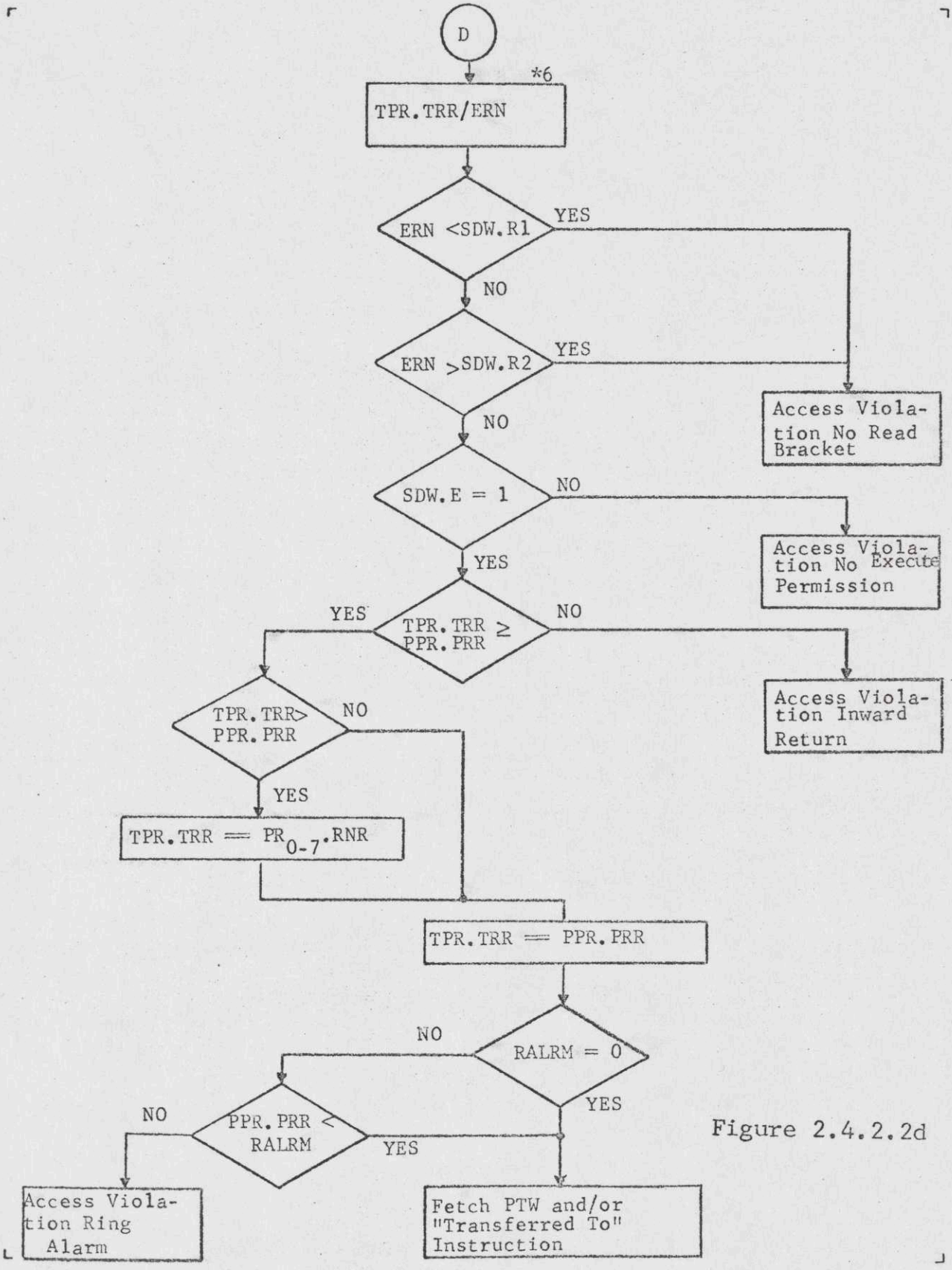


Figure 2.4.2.2d

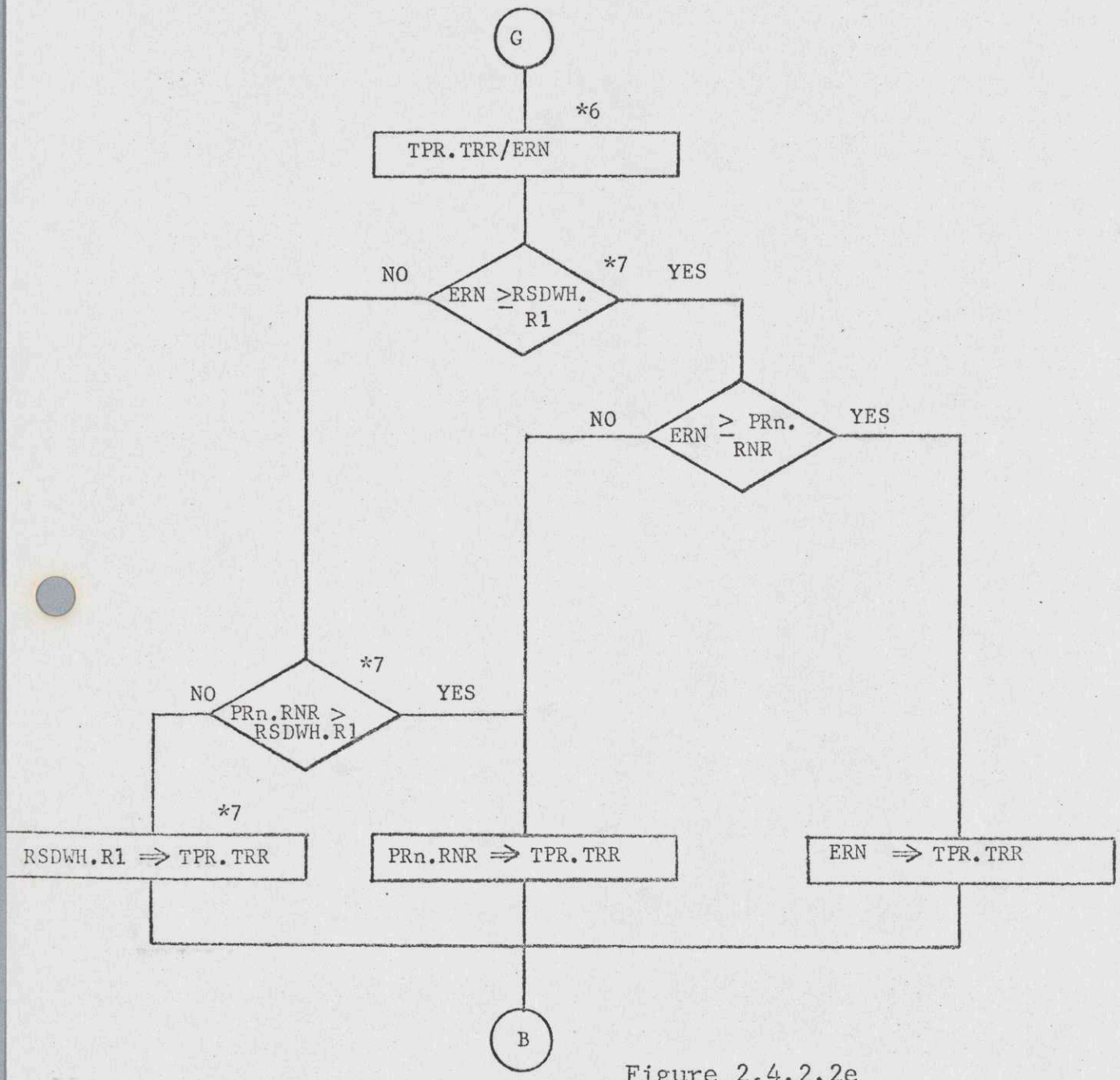


Figure 2.4.2.2e

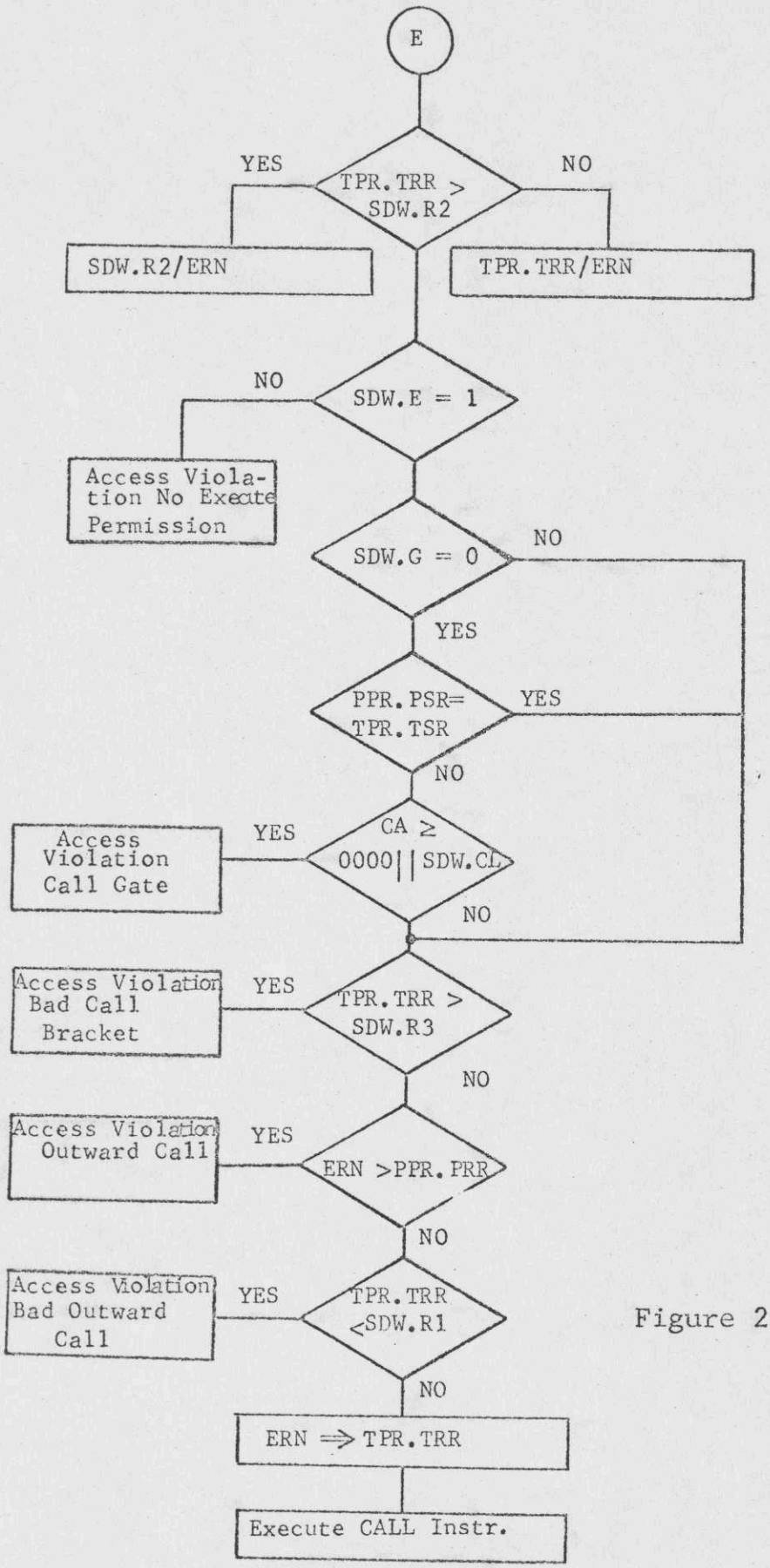


Figure 2.4.2.2e

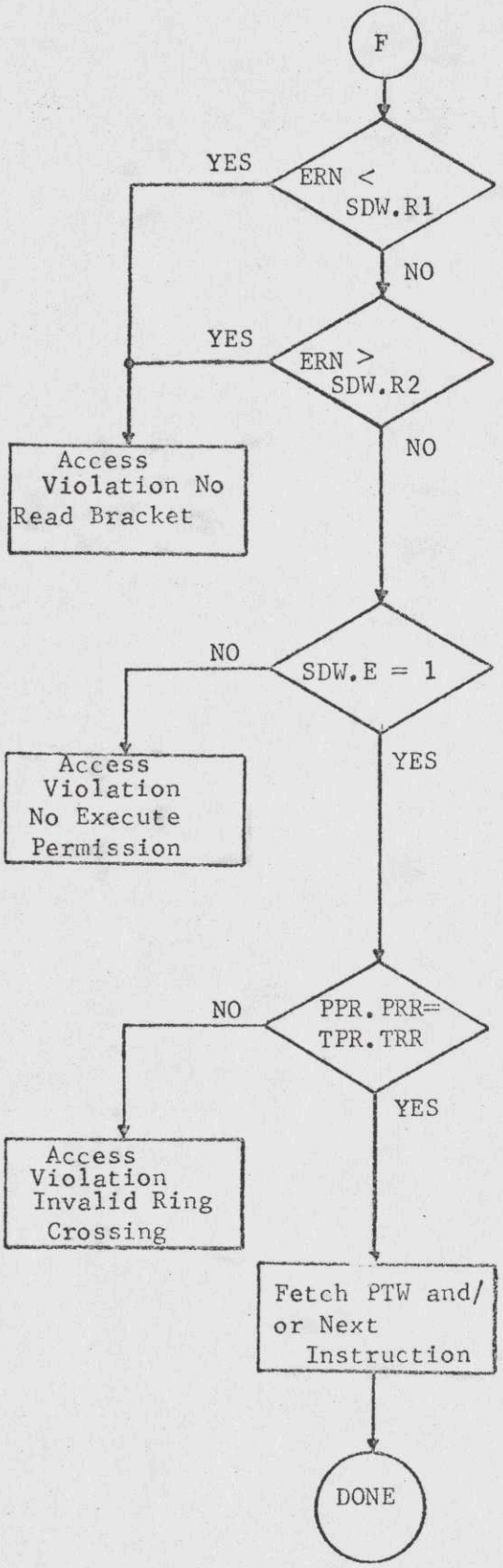


Figure 2.4.2.2f

2.5 INSTRUCTION MODIFICATION OF APPENDING UNIT REGISTERS IN MULTICS OPERATIONAL MODE

2.5.1 Data Movement Instructions

These instructions shall modify the appending unit registers as specified in Section 3.4.1.

2.5.2 CALL Instruction

Following the address preparation and access verification, the CALL instruction shall modify Pointer Register #7 (PR_7) and the Procedure Pointer Register (PRR) as shown in Figure 2.5.2.

If the CALL instruction is executed in the Absolute Address Mode without bit 29 equal to a "1" and without indirection through an ITS pair, the following shall apply:

a. The Append Address Mode shall be entered for the address preparation of the call address and if the CALL executes successfully the Append Address Mode shall be retained.

b. The Effective Segment Number (ESN) generated for the SDW fetch and subsequent loading into TPR.TSR prior to the call execution (Figure 2.5.2a) shall be equal to zero.

c. The Effective Ring Number (ERN) loaded into TPR.TRR prior to the SDW fetch shall be equal to zero which implies that the access check for an outward call will not fault and the bad call bracket check will fault if SDW.R1 is not equal to zero. (See Figures 2.4.2.2e and 2.5.2.)

2.5.3 RTCD Instruction

Following the fetch of the RTCD operand pair, the TPR.TRR register shall be loaded as shown in Section 2.4.2.2 and:

a. $C(Y_{03-17}) \Rightarrow$ TPR.TSR

b. $C(Y+1_{00-17})$ = TPR.CA

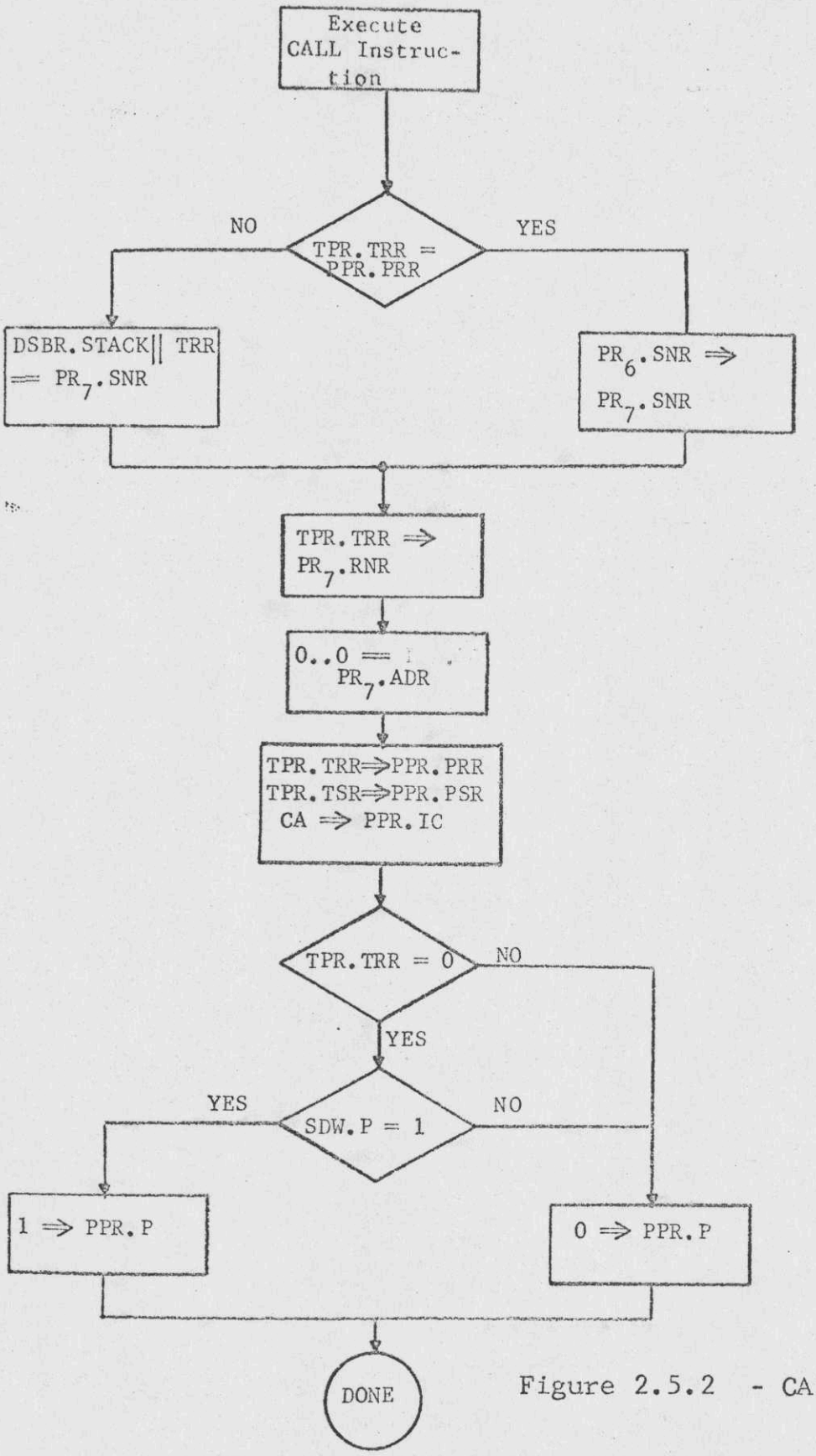


Figure 2.5.2 - CALL Instruction Execution

Following the address preparation and access verification for the "transferred to" segment, the RTCD instruction shall cause the complete Procedure Pointer Register PPR to be loaded peior to checking for a ring alarm access violation. The loading of PPR shall be:

TPR.TSR ⇒ PPR.PSR

TPR.TRR ⇒ PPR.PRR

TPR.CA ⇒ PPR.IC

SDW.P ⇒ PPR.P (if TPR.TRR = 000)

"0" ⇒ PPR.P (if TPR.TRR ≠ 000)

If the RTCD instruction is executed in the Absolute Address Mode without bit 29 equal to a "1" and without encountering an ITS or ITP pair, the following shall occur:

a. The Append Address Mode shall be entered for the address preparation of the RTCD operand and this mode shall be retained if the RTCD executes successfully.

b. The Effective Segment Number (ESN) generated to fetch the SDW for the RTCD operand and then loaded into TPR.TSR shall be equal to the previous value of PPR.PSR.

c. The Effective Ring Number (ERN) loaded into TPR.TRR prior to the access checks for the RTCD operand shall be equal to zero.

2.5.4 RCU Instruction

The RCU shall load the appending unit registers as specified in Section 3.4.2.

2.5.5 TSS Instruction

Following the address preparation and normal transfer access checking the TSS instruction shall cause the PPR to be loaded as specified in Section 2.4.2.2. The PPR.P bit is set to a one if TPR.TRR = 000 and SDW.P = 1, otherwise it is reset.

If the TSS instruction is executed in the Absolute Address Mode without bit 29 equal to a "1" and without indirection through an ITS or TTP pair, the following shall apply.

a. The Append Address Mode shall be entered for the address preparation of the transfer address and this mode shall be retained if the TSS executes successfully.

b. The Effective Segment Number (ESN) generated for the SDW fetch and subsequent loading into TPR.TSR shall be the previous valud of PPR.PSR.

c. The Effective Ring Number (ERN) loaded into TPR.TRR prior to the access checks for the instruction shall be zero.

If a TSS instruction is executed in "Ring 0" to a privileged segment (SDW.P = 1) the privileged bit in the PPR shall be set to a one. Subsequent, attempted executions of 645F privileged instructions will be trapped as illegal op-code faults rather than as privileged instructions faults.

2.5.6 All Other Transfer Instructions

Except for the instructions described above all transfer instructions executed in the Append Address Mode shall cause the PPR register to be loaded following access checks for the transferred to instruction as follows:

TPR.TSR $\Rightarrow$ PPR.PSR

TPR.CA $\Rightarrow$ PPR.IC

SDW.P $\Rightarrow$ PPR.P (if TRR = 000)

0 $\Rightarrow$ PPR.P (if TRR $\neq$ 000)

2.5.7 The ITS Modifier

When the processor accesses an even indirect word due to the RI or IR tag in an instruction or indirect word, and this indirect word contains the ITS modifier tag, the contents of this indirect word shall be loaded as follows:

$C(\text{ITS Word})_{03\text{-}17} \Rightarrow \text{TPR.TSR}$

$C(\text{ITS Word})_{18\text{-}20} \Rightarrow \text{RIB}$

$C(\text{ITS Word} + 1)_{00\text{-}17} + C(R) \Rightarrow \text{TPR.CA}$

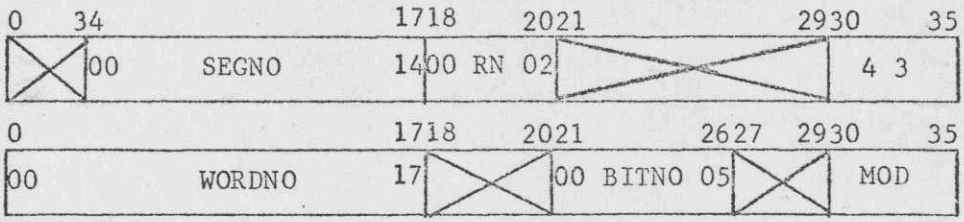
$C(\text{ITS Word} + 1)_{21\text{-}26} \Rightarrow \text{TPR.TBR}$

where C(R) is specified by the previous IR tag or the tag field of ITS Word + 1

TPR.TRR is loaded as specified in 2.4.2.2.

For all further indirection following an ITS pair the Effective Segment Number (ESN) used shall be derived from TPR.TSR and the Effective Ring Number (ERN) used shall equal to the ERN generated following the receipt of the ITS pair as specified in 2.4.2.2.

The format of the ITS word pair in memory shall be:



where: SEGNO = Indirect Segment Number

RN = Maximum level of privilege which can be assigned to the current procedure segment for accessing the specified segment.

43_8 = ITS designator

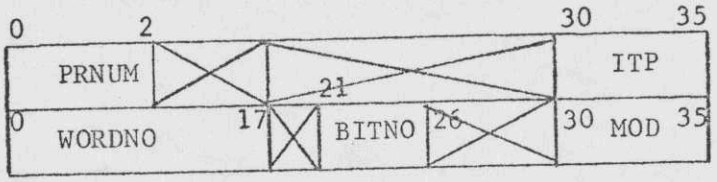
WORDNO = Word offset to be used in generating CA for the indirect segment access.

BITNO = Bit offset

MOD = Tag to specify further indirection

2.5.8 ITP Modifier

In the 645F Processor the indirect word for ITP modification shall consist of the following format:



PRNUM (0-2) is the number of the pointer register to use in the indirection,

ITP (30-35) is the "indirect through pointer register" modification (41)8,

WORDNO (0-17) is the relative offset to be used in the effective address formation as described below,

BITNO (21-26) is the bit offset to be used as described below,

MOD (30-35) is any valid

When an ITP modifier is found in the (even) word of an indirect reference the temporary pointer register (TPR) is loaded as follows:

$C(PRn.ADR)_{0-17} + C(ITP.WORDNO) + C(R) \Rightarrow TPR.CA$

$C(PRn.SNR) \Rightarrow C(TPR.TSR)$

$C(ITP.BITNO) \Rightarrow C(TPR.TBR)$

$MAX(PRn.RNR, SDW.R1, TPR.TRR) \Rightarrow C(TPR.TRR)$

where n = ITP.PRNUM. Further indirection is possible if ITP.MOD so indicates.

C(R) is specified by the previous IR tag or the MOD field of IPT word + 1.

3. CUSTOMER INTERFACE

3.1 PROGRAM COMPATIBILITY

The 645F Processor shall have all the instructions, modifiers, and fault provisions of the H6000 Processor, in addition to instructions defined as exclusive 645F instructions. Any operation code previously used in the 605, 615, 635 or 645, shall not be used as a new operation code unless the two instructions perform the same function in each machine.

Figure 3.1 shows the operation code map with bit 27 = 0.

Figure 3.2 shows the operation code map with bit 27 = 1.

The 645F Processor instructions are indicated by being in parenthesis and attempts to execute them in the GECOS operational mode or while in the BAR Mode shall result in an illegal procedure fault.

The privileged mode instructions are indicated by being underlined and attempts to execute them in the unprivileged or slave modes shall result in a command fault in the GECOS operational mode and an IPR fault in the MULTICS operational mode.

3.2 DATA FORMATS

The 645F Processor shall be capable of addressing data in all the formats that the H6000 Processor can - 1, 4, 6, 9, 18, 36 and 72-bit. (See Section 3.2 of the 655/EIS Processor EPS-1 - 43A232719.)

	000	001	002	003	004	005	006	007
000		MME	DRL		(MME2)	(MME3)		(MME4)
020	ADLX0	ADLX1	ADLX2	ADLX3	ADLX4	ADLX5	ADLX6	ADLX7
040	ASX0	ASX1	ASX2	ASX3	ASX4	ASX5	ASX6	ASX7
060	ADX0	ADX1	ADX2	ADX3	ADX4	ADX5	ADX6	ADX7
100	CMPX0	CMPX1	CMPX2	CMPX3	CMPX4	CMPX5	CMPX6	CMPX7
120	SBLX0	SBLX1	SBLX2	SBLX3	SBLX4	SBLX5	SBLX6	SBLX7
140	SSX0	SSX1	SSX2	SSX3	SSX4	SSX5	SSX6	SSX7
160	SBX0	SBX1	SBX2	SBX3	SBX4	SBX5	SBX6	SBX7
200	CNAX0	CNAX1	CNAX2	CNAX3	CNAX4	CNAX5	CNAX6	CNAX7
220	LDX0	LDX1	LDX2	LDX3	LDX4	LDX5	LDX6	LDX7
240	ORSX0	ORSX1	ORSX2	ORSX3	ORSX4	ORSX5	ORSX6	ORSX7
260	ORX0	ORX1	ORX2	ORX3	ORX4	ORX5	ORX6	ORX7
300	CANX0	CANX1	CANX2	CANX3	CANX4	CANX5	CANX6	CANX7
320	LCX0	LCX1	LCX2	LCX3	LCX4	LCX5	LCX6	LCX7
340	ANSX0	ANSX1	ANSX2	ANSX3	ANSX4	ANSX5	ANSX6	ANSX7
360	ANX0	ANX1	ANX2	ANX3	ANX4	ANX5	ANX6	ANX7
400		MPF	MPY			CMG		
420		UFM		DUFM		FCMG		DFCMG
440	SXL0	SXL1	SXL2	SXL3	SXL4	SXL5	SXL6	SXL7
460		FMP		DFMP				
500	RPL					BCD	DIV	DVF
520	RPT					FDI		DFDI
540	(SPRP0)	(SPRP1)	(SPRP2)	(SPRP3)	(SPRP4)	(SPRP5)	(SPRP6)	(SPRP7)
560	RPD					FDV		DFDV
600	TZE	TNZ	TNC	TRC	TMI	TPL		TTF
620	EAX0	EAX1	EAX2	EAX3	EAX4	EAX5	EAX6	EAX7
640	ERSX0	ERSX1	ERSX2	ERSX3	ERSX4	ERSX5	ERSX6	ERSX7
660	ERX0	ERX1	ERX2	ERX3	ERX4	ERX5	ERX6	ERX7
700	TSX0	TSX1	TSX2	TSX3	TSX4	TSX5	TSX6	TSX7
720	LXL0	LXL1	LXL2	LXL3	LXL4	LXL5	LXL6	LXL7
740	STX0	STX1	STX2	STX3	STX4	STX5	STX6	STX7
760	(LPRP0)	(LPRP1)	(LPRP2)	(LPRP3)	(LPRP4)	(LPRP5)	(LPRP6)	(LPRP7)

Figure 3.1a - Operation Code Map Bit 27 = 0
(Left Half)

010		011	012	013	014	015	016	017
000		NOP				CIOC		
020			LDQC	ADL	LDAC	ADLA	ADLQ	ADLAQ
040	(ADWP0)	(ADWP1)	(ADWP2)	(ADWP3)	AOS	ASA	ASQ	SSCR
060		AWCA	AWCQ	LREG		ADA	ADQ	ADAQ
100		CWL				CMPA	CMPQ	CMPAQ
120						SBLA	SBLQ	SBLAQ
140	(ADWP4)	(ADWP5)	(ADWP6)	(ADWP7)	(SDBR)	SSA	SSQ	
160		SWCA	SWCQ	(LPRI)		SBA	SBQ	SBAQ
200		CMX	(EPAQ)	(ABSA)	SZNC	CNAA	CNAQ	CNAAQ
220	LBAR	RSW*	(LDBR)	RMCM	SZN	LDA	LDQ	LDAQ
240	(SPRI0)	(SPBP1)	(SPRI2)	(SPBP3)	(SPRI)	ORSA	ORSQ	(LSDP)
260	(TSP0)	(TSP1)	(TSP2)	(TSP3)		ORA	ORQ	ORAQ
300	(EQWP0)	(EASP0)	(EAWP2)	(EASP2)		CANA	CANQ	CANAQ
320	(EAWP4)	(EASP4)	(EAWP6)	(EASP6)		LCA	LCQ	LCAQ
340	(EPP0)	(EPBP1)	(EPP2)	(EPBP3)	(STAC)	ANSA	ANSQ	(STCD)
360	(EPP4)	(EPBP5)	(EPP6)	(EPBP7)		ANA	ANQ	ANAQ
400		LDE		RSCR		ADE		
420	FSZN	FLD		DFLD		UFA		DUFA
440	STZ	SMIC	SCPR		STT	FST	STE	DFST
460	FSTR	FRD	DFSTR	DFRD		FAD		DFAD
500				FNEG		FCMP		DFCMP
520		NEG	(CAMS)	NEGL		UFS		DUFS
540	SBAR	STBA	STBQ	SMCM	STC1			(SSDP)
560				FNO		FSB		DFSB
600	(RTCD)			(RCU)	TEO	TEU	DIS	TOV
620	RET			(RCCL)	LDI	EAA	EAQ	LDT
640	(SPRI4)	(SPBP5)	(SPBI6)	(SPBP7)		ERSA	ERSQ	(SCU)
660	(TSP4)	(TSP5)	(TSP6)	(TSP7)	LCPR	ERA	ERQ	ERAQ
700	TRA			(CALL)		TSS	XEC	XED
720		ARS	QRS	LRS		ALS	QLS	LLS
740	STC2	STCA	STCQ	SREG	STI	STA	STQ	STAQ
760		ARL	QRL	LRL	GTB	ALR	QLR	LLR

Figure 3.1b - Operation Code Map Bit 27 = 0
(Right Half)

*RSW shall not be listed in any Programming Reference Manual

	000	001	002	003	004	005	006	007
000								
020	MVE				MVNE			
040								
060	CSL	CSR			SZTL	SZTR	CMPB	
100	MLR	MRL					CMPC	
120	SCD	SCDR			SCM	SCMR		
140								
160	MVT				TCT	TCTR		
200			AD2D	SB2D			MP2D	DV2D
220			AD3D	SB3D			MP3D	DV3D
240								
260								
300	MVN	BTD		CMPN		DTB		
320								
340								DVDR
360								
400								
420								
440				SAREG				SPL
460				LAREG				LPL
500	A9BD	A6BD	A4BD	ABD				AWD
520	S9BD	S6BD	S4BD	SBD				SWD
540	ARA0	ARA1	ARA2	ARA3	ARA4	ARA5	ARA6	ARA7
560	AAR0	AAR1	AAR2	AAR3	AAR4	AAR5	AAR6	AAR7
600	TRTN	TRTF	TPNZ	TTN				
620								
640	ARN0	ARN1	ARN2	ARN3	ARN4	ARN5	ARN6	ARN7
660	NAR0	NAR1	NAR2	NAR3	NAR4	NAR5	NAR6	NAR7
700								
720								
740	SAR0	SAR1	SAR2	SAR3	SAR4	SAR5	SAR6	SAR7
760	LAR0	LAR1	LAR2	LAR3	LAR4	LAR5	LAR6	LAR7

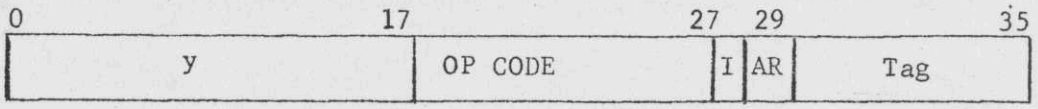
Figure 3.2a - Operation Code Map Bit 27 = 1
(Left Half)

	010	011	012	013	014	015	016	017
000 020 040 060								
100 120 140 160				(LPTR)	(SPTR)			
200 220 240 260	(SPBP0)	(SPRI1)	(LSDR) (SPBP2)	(SPRI3)	(SSDR)			(LPTP)
300 320 340 360	(EASP1) (EASP5) (EPBP0) (EPBP4)	(EAWP1) (EAWP5) (EPP1) (EPP5)	(EASP3) (EASP7) (EPBP2) (EPBP4)	(EAWP3) (EAWP7) (EPP3) (EPP7)				
400 420 440 460								
500 520 540 560			(CAMP)					(SPTP)
600 620 640 660	(SPBP4)	(SPRI5)	(SPBP6)	(SPRI7)				
700 720 740 760					(SRA) (LRA)		(SPTR)	

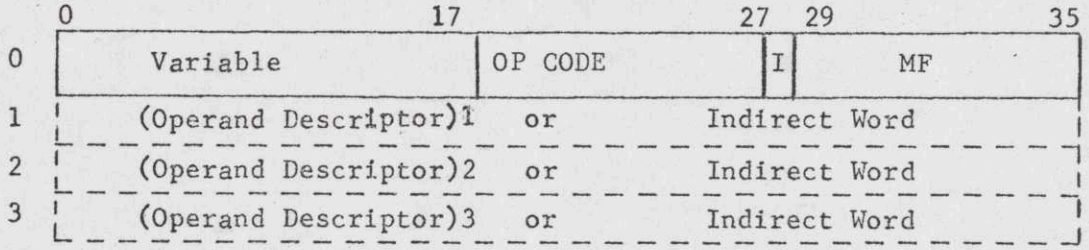
Figure 3.2b - Operation Code Map Bit 27 = 1
(Right Half)

3.3 INSTRUCTION WORD FORMAT

The 645F Processor shall recognize both the standard H6000 Instruction Format(with extended op code field):

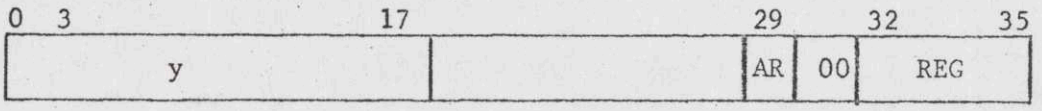


and the EIS multiword instruction format:

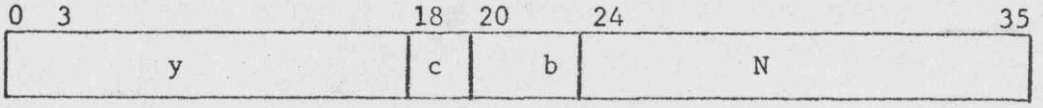


where:

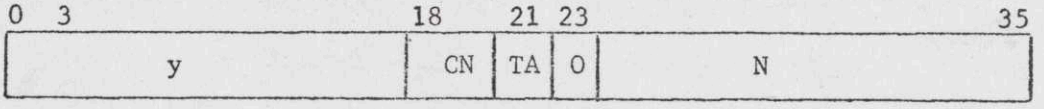
indirect word format:



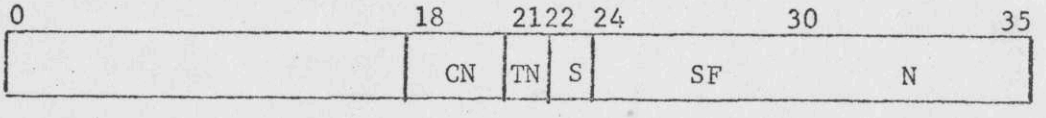
and bit string operand:



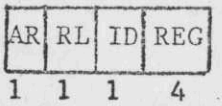
Alphanumeric operand:



Numeric operand:



and modification fields format:



For field definitions, see Section 3.3.3 of 655 EIS Processor, EPS-1 43A232719.

In the MULTICS Operational Mode, address preparation shall be performed as described in Section 2.3. In the GECOS Operational Mode, address preparation shall be performed as described in Section 3.3.4 of the 655/EIS Processor EPS (43A232719).

3.4 INSTRUCTION DESCRIPTIONS

In the GECOS Operational Mode, the 645F Processor shall execute all the instructions described in the 655 EIS Processor EPS-1 (43A232719). All other op codes (including legal 645F instruction op codes) will be illegal and attempted execution shall cause an Illegal Procedure Fault (IPR).

In the MULTICS Operational Mode, the 645F Processor shall execute all the instructions described in the 655 EIS Processor EPS-1 (43A232719), and the following 645F instructions:

3.4.1 Data Movement Instructions

Mnemonic	Name of Instruction	Op Code(octal)
$LPRP_n$	Load Pointer Register n Packed	76n (0)

Summary: $C(Y)_{0-35} \Longrightarrow C(PRn)$ where:

$C(TPR.TRR) \Longrightarrow C(PRn.RNR)$

$C(Y)_{0-5} \xrightarrow{b/cb} C(PRn.ADR)_{18-23}$

$*1 || C(Y)_{6-17} \Longrightarrow C(PRn.SNR)$

$C(Y)_{18-35} \Longrightarrow C(PRn.ADR)_{0-17}$

*1 The $C(PRn.SNR)_{0-2}$ will be replaced with ones if the $C(Y)_{6-17}$ is ones. If the $C(Y)_{6-17}$ is not all ones, the $C(PRn.SNR)_{0-2}$ will be replaced with zeros.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Faults:
Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
LPRI	Load Pointer Registers from ITS Pair	173 (0)

Summary: $C(Y, Y+2,.....Y+14)_{00-71} \Longrightarrow C(PR0,PR1....PR7$

where: MAX(Y+2n pair 18-20,SDW.R1,TPR.TRR) $\Longrightarrow$ C(PRn.RNR)

$C(Y + 2n\ pair)_{3-17} \Longrightarrow C(SNRn)$

$C(Y + 2n\ pair)_{36-53} \Longrightarrow C(ADRn)_{00-17}$

$C(Y + 2n\ pair)_{57-62} \xrightarrow{b/cb} C(ADRn)_{18-23}$

Starting at location Y, the contents of eight word pairs (in ITS pair format) replace the contents of Pointer Registers 0 through 7 as shown. The hardware shall assume Y_{14-17}=0000 (zero modulo 16) and addressing will be incremented accordingly. No check shall be made. The contents of word pairs loaded shall be unchanged.

Modification: All except DU, DL, C, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Note: This replaces 645 Load Base Register instruction.

Mnemonic	Name of Instruction	Op Code(octal)
LDBR	Load Descriptor Segment Base Register	232 (0)

Summary: $0 \Longrightarrow C(AM.FULL)$ (No effect if AM is disabled by switch or error)

$0000\text{-}1111 \Longrightarrow C(AM.USE)$

$C(Y)_{0\text{-}71} \Longrightarrow C(DSBR)$

$C(Y\ pair)_{0\text{-}23} \Longrightarrow C(DSBR.ADDR)$

$C(Y\ pair)_{37\text{-}50} \Longrightarrow C(DSBR.BOUND)$

$C(Y\ pair)_{55} \Longrightarrow C(DSBR.U)$

$C(Y\ pair)_{60\text{-}71} \Longrightarrow C(DSBR.STACK)$

The contents of the word pair starting at location Y replace the contents of the Descriptor Segment Base Register as shown. The hardware shall assume $Y_{17} = 0$. No check shall be made. Both halves of the associative memory is cleared (FULL indicators reset). The contents of location Y shall be unchanged.

Modifications: All except DU, DL, CI, SC, SCR.

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode.

Mnemonic	Name of Instruction	Op Code (octal)
LRA	Load Ring Alarm	774 (1)

Summary: $C(Y)_{33-35} \Longrightarrow C(RALR)$

The contents of location Y, bits 33-35, replace the contents of the Ring Alarm Register. The contents of location Y remain unchanged.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, SC, SCR, CI
- Attempted execution in the unprivilaged mode.

Mnemonic	Name of Instruction	Op Code (octal)
LSDR	Load Segment Descriptor Register	232 (1)

Summary: $C(Y,Y+2),....Y+30)_{0-71} \Longrightarrow C(AM.SDR)$ where:

$C(Y+2m)_{0-23} \Longrightarrow C(SDR.ADDR)$

$C(Y+2m)_{24-32} \Longrightarrow C(SDR.R1,R2,R3)$

$C(Y+2m)_{37-50} \Longrightarrow C(SDR.BOUND)$

$C(Y+2m)_{51-56} \Longrightarrow C(SDR.R,E,W,P,U,G)$

$C(Y+2m)_{58-71} \Longrightarrow C(SDR.CL)$

} where $C(AM.SDW.USE) = m$

As defined by m, the contents of the USE field of each segment descriptor word associative memory register; the contents of the 16 word pairs starting at location Y replace the segment descriptor word portion of the AM-SDW as shown. The hardware shall assume $Y_{13-17} = 00000$, and addressing will be incremented by 2m accordingly. No check shall be made. The rest of the register shall be unchanged. The associative memory is ignored (forced to a "no match") during the address generation phase.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
LPTR	Load Page Table Register	173 (1)

Summary: $C(Y, Y+1,....Y+15)_{0-35} \Longrightarrow C(AM.PTR)$ where:

$$\left.\begin{array}{l} C(Y+m)_{0-17} \Longrightarrow C(PTR.ADDR) \\ C(Y+m)_{29} \Longrightarrow C(PTR.M) \end{array}\right\} \text{where } C(AM.PTW.USE)_{=m}$$

As defined by m, the contents of the USE field of each page table word associative memory register; contents of location Y+m replaces contents of the page table register of the AM-PTW as shown. The hardware shall assume $Y_{14-17} = 0000$ and addressing will be incremented by m accordingly. No check shall be made. The rest of the register shall be unchanged. The associative memory is ignored (forced to a "no match") during the address generation phase.

Modifications; All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
LPTP	Load Page Table Pointers	257 (1)

Summary: $C(Y,Y+1, \ldots Y+15)_{0-35} \Longrightarrow C(AM.PTP)$ where:

$$\left.\begin{array}{l} C(Y+m)_{0-14} \Longrightarrow C(PTP.POINTER) \\ C(Y+m)_{15-26} \Longrightarrow C(PTP.PAGE\ NO.) \\ C(Y+m)_{27} \Longrightarrow C(PTP.F) \end{array}\right\} \text{where } C(AM\text{-}PTW.USE) = m$$

According to m, the contents of the USE field of each page table word associative memory register; the contents of location Y+m replaces the POINTER, PAGENO, and full/empty bit of the AM-PTW as shown. The hardware shall assume $Y_{14-17} = 0000$. No check shall be made. The rest of the register shall be unchanged. The associative memory is ignored (forced to "no match") during the address generation phase.

Modification: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
LSDP	Load Segment Descriptor Pointers	257 (0)

Summary: $C(Y, Y+1, \ldots Y+15)_{0-35} \Longrightarrow C(AM.SDP)$ where:

$$\left.\begin{aligned} C(Y+m)_{0-14} &\Longrightarrow C(SDP.POINTER) \\ C(Y+m)_{27} &\Longrightarrow C(SDP.F) \end{aligned}\right\} \text{where } C(AM\text{-}SDW.USE) = m$$

As defined by m, the contents of the USE field of each segment descriptor word associative memory register; the contents of location Y+m replaces the POINTER and full/empty bit of the AM-SDW as shown. The hardware shall assume $Y_{14-17} = 0000$. No check shall be made. The rest of the register shall be unchanged. The associative memory is ignored (forced to "no match") during the address generation phase.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- o Modifications: DU, DL, CI, SC, SCR
- o Attempted execution in the unprivileged mode

Note: This and LPTR replaces 645 Load Associative Memory instruction.

Mnemonic	Name of Instruction	Op Code (octal)
EASPn	Effective Address to Segment Number Field of PRn	311 (0) 310 (1) 313 (0) 312 (1) 331 (0) 330 (1) 333 (0) 332 (1)

Summary: $EA_{3-17} \Longrightarrow C(PRn.SNR)$

The least significant 15 bit of the effective address register replaces the contents of the SEGNO field of Pointer Register n. The rest of Pointer Register n remains unchanged.

Modification: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modification: DU, DL, CI, SC, SCR

Note: This and EAWPn replace 645 Effective Address to Base n instruction.

Mnemonic	Name of Instruction	Op Code (octal)
EAWPn	Effective Address to Word/ Bit Number Field of PRn	310 (0) 311 (1) 312 (0) 313 (1) 330 (0) 331 (1) 332 (0) 333 (1)

Summary: $C(TPR.CA) \Longrightarrow C(PRn.ADR)_{0-17}$

$C(TPR.TBR) \underline{b/cb} \Rightarrow C(PRn.ADR)_{18-23}$

The calculated address and bit replaces the contents of the word and bit fields of Pointer Register n. The remaining fields of Pointer Register n remain unchanged.

Modification: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modification: DU, DL, CI, SC, SCR

Note: This and EAWPn replace 645 Effective Address to Base n instruction.

Mnemonic	Name of Instruction	Op Code (octal)
EPPn	Effective Pointer to PRn	350 (0) 351 (1) 352 (0) 353 (1) 370 (0) 371 (1) 372 (0) 373 (1)

Summary: C(TPR) $\Longrightarrow$ C(PRn)

C(TPR.TRR) $\Longrightarrow$ C(PRn.RNR)

C(TPR.TSR) $\Longrightarrow$ C(PRn.SNR)

C(TPR.CA) $\Longrightarrow$ C(PRn.ADR)$_{0-17}$

C(TPR.TBR) $\xrightarrow{b/cb}$ CPRn.ADR)$_{18-23}$

The ring number, segment number, bit number and calculated address generated during the address preparation sequence shall correspondingly replace the contents of the RN,SEGNO,WORD and BIT fields of Pointer Register n.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modification: DU, DL, CI, SC, SCR

Note: This and EPBPn replace 645 Effective Address to Pair n instruction.

Mnemonic	Name of Instruction	Op Code (octal)
EPBPn	Effective Pointer at Base to PRn	350 (1) 351 (0) 352 (1) 353 (0) 370 (1) 371 (0) 372 (1) 373 (0)

Summary: $C(TPR.TRR) \Longrightarrow C(PRn.RNR)$

$C(TPR.TSR) \Longrightarrow C(PRn.SNR)$

$00 \ldots\ 00 \Longrightarrow C(PRn.ADR)_{00-17}$

$00 \ldots\ 00 \Longrightarrow C(PRn.ADR)_{18-23}$

The ring number and segment descriptor number currently stored in the temporary pointer register shall correspondingly replace the contents of the RN and SEGNO fields of Pointer Register n. The contents of the WORD $(ADR)_{00-17}$ and BIT $(ADR)_{18-23}$ fields of Pointer Register n are forced to zero. The contents of the temporary pointer register are unchanged.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modification: DU, DL, CI, SC, SCR

Note: This and EPPn replace 645 Effective Address to Pair n instruction.

Mnemonic	Name of Instruction	Op Code (octal)
EPAQ	Effective Pointer to AQ Register	213 (0)

Summary: $C(TPR) \Longrightarrow C(AQ)_{00-71}$

where:

$00...00 \Longrightarrow C(AQ)_{0-2}$

$C(TPR.TSR) \Longrightarrow C(AQ)_{3-17}$

$00...00 \Longrightarrow C(AQ)_{18-32}$

$C(TPR.TRR) \Longrightarrow C(AQ)_{33-35}$

$C(TPR.CA) \Longrightarrow C(AQ)_{36-53}$

$00...00 \Longrightarrow C(AQ)_{54-65}$

$C(TPR.TBR) \Longrightarrow C(AQ)_{66-71}$

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected:

Zero	If C(AQ) = 0, then ON; otherwise OFF
Negative	If $C(AQ)_0$ = 1, then ON: otherwise OFF

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
ABSA	Absolute Address to Accumulator	212 (0)

Summary: OPERAND ADDRESS $\Longrightarrow C(A)_{0-23}$

00 ..... 00 $\Longrightarrow C(A)_{24-35}$

The 24 bit operand address replaces the most significant 24 bits of the accumulator. Zeros replace the least significant 12 bits of the accumulator. If this instruction is executed in the Absolute Addressing Mode the data in the accumulator is defined.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected:

Zero	If C(A) = 0, then ON: otherwise OFF
Negative	If $C(A)_0 = 1$, then ON; otherwise OFF

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode.

Mnemonic	Name of Instruction	Op Code (octal)
SPRPn	Store Pointer Register n Packed	54n (0)

Summary: $C(PRn) \Longrightarrow C(Y)_{0-35}$

$C(ADRn)_{18-23}$ cb/b $\Rightarrow C(Y)_{0-5}$

$C(SNRn)_{3-14} \Longrightarrow C(Y)_{6-17}$

$C(ADRn)_{0-17} \Longrightarrow C(Y)_{18-35}$

The contents of Pointer Register n replace the contents of location Y as shown. The contents of Pointer Register n shall be unchanged.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modification: DU, DL, CI, SC, SCR

Store Fault:

- If 3 most significant bits of C(PRn·SNR) ≠ 000 and C(PRn·SNR) ≠ to all one's (Null Pointer).

Note: This replaces 645 Store Address Base Register n instruction.

Mnemonic	Name of Instruction	Op Code (octal)
SPRI	Store Pointer Registers as ITS Pairs	254 (0)

Summary: $C(PR0, PR1, \ldots PR7) \Longrightarrow C(Y,Y+2\ldots Y+14)_{0-71}$

where:

$000 \Longrightarrow C(Y+2n\ pair)_{0-2}$

$C(PRn.SNR) \Longrightarrow C(Y+2n\ pair)_{3-17}$

$C(PRn.RNR) \Longrightarrow C(Y+2n\ pair)_{18-20}$

$00\ldots00 \Longrightarrow C(Y+2n\ pair)_{21-29}$

$43_8 \Longrightarrow C(Y+2n\ pair)_{30-35}$

$C(PRn.ADR)_{0-17} \Longrightarrow C(Y+2n\ pair)_{36-53}$

$0\ 0\ 0 \Longrightarrow C(Y+2n\ pair)_{54-56}$

$C(PRn.ADR)_{18-23} \xrightarrow{cb/b} C(Y+2n\ pair)_{57-62}$

$00\ldots00 \Longrightarrow C(Y+2n\ pair)_{63-71}$

Starting at location Y, the contents of Pointer Registers 0-7 replace the contents of eight word pairs (in ITS pair format). The hardware shall assume $Y_{14-17} = 0000$ and addressing will be incremented accordingly. No check shall be made. The contents of the Pointer Registers shall be unchanged.

Modifications: All except DU, CL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Note: This replaces 645 Store Base instruction.

Mnemonic	Name of Instruction	Op Code (octal)
SPRIn	Store Pointer Register n as ITS Pair	250 (0) 251 (1) 252 (0) 253 (1) 650 (0) 651 (1) 652 (0) 653 (1)

Summary: $C(PRn) \Longrightarrow C(Y)_{0-71}$

where:

$000 \Longrightarrow C(Y\ pair)_{0-2}$

$C(PRn.SNR) \Longrightarrow C(Y\ pair)_{3-17}$

$C(PRn.RNR) \Longrightarrow C(Y\ pair)_{18-20}$

$00\ \ldots\ 00 \Longrightarrow C(Y\ pair)_{21-29}$

$43_8 \Longrightarrow C(Y\ pair)_{30-35}$

$C(PRn.ADR)_{0-17} \Longrightarrow C(Y\ pair)_{36-53}$

$000 \Longrightarrow C(Y\ pair)_{54-56}$

$C(PRn.ADR)_{18-23} \xrightarrow{cb/b} C(Y\ pair)_{57-62}$

$00\ \ldots\ 00 \Longrightarrow C(Y\ pair)_{63-71}$

The contents of Pointer Register n replaces the contents of the word pair (in ITS format) starting at location Y. The hardware shall assume $Y_{17} = 0$. No check shall be made. The contents of the Pointer Register n shall be unchanged.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:
Modifications: DU, DL, CI, SC, SCR
Note: This and SPBPn replace 645 Store Pair n instruction.

Mnemonic	Name of Instruction	Op Code (octal)
SPBPn	Store Segment Base Pointer of PRn	250 (1) 251 (0) 252 (1) 253 (0) 650 (1) 651 (0) 652 (1) 653 (0)

Summary: $000 \Longrightarrow C(\text{Y pair})_{0\text{-}2}$

$C(\text{PRn.SNR}) \Longrightarrow C(\text{Y pair})_{3\text{-}17}$

$C(\text{PRn.RNR}) \Longrightarrow C(\text{Y pair})_{18\text{-}20}$

$00 \ldots 00 \Longrightarrow C(\text{Y pair})_{21\text{-}29}$

$43_8 \Longrightarrow C(\text{Y pair})_{30\text{-}35}$

$00 \ldots 00 \Longrightarrow C(\text{Y pair})_{36\text{-}71}$

The ring number and segment descriptor number currently stored in the Pointer Register n replace the contents of word pair starting at location Y as shown. The hardware shall assume $Y_{17} = 0$. No check shall be made. The contents of Pointer Register n shall be unchanged.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Note: This and SPIPn replace 645 Store Pair n instruction.

Mnemonic	Name of Instruction	Op Code (octal)
SDBR	Store Descriptor Segment Base Register	154 (0)

Summary: $C(DSBR) \Longrightarrow C(Y)_{0-71}$

where:

$C(DSBR.ADDR) \Longrightarrow C(Y\ pair)_{0-23}$

$00\ .\ .\ .\ 00 \Longrightarrow C(Y\ pair)_{24-36}$

$C(DSBR.BOUND) \Longrightarrow C(Y\ pair)_{37-50}$

$000 \Longrightarrow C(Y\ pair)_{51-54}$

$C(DSBR.U) \Longrightarrow C(Y\ pair)_{55}$

$000 \Longrightarrow C(Y\ pair)_{56-59}$

$C(DSBR.STACK) \Longrightarrow C(Y\ pair)_{60-71}$

The contents of the Descriptor Segment Base Register replace the contents of the word pair starting at location Y as shown. The hardware shall assume $Y_{17} = 0$. No check shall be made. The contents of the Descriptor Segment Base Register shall be unchanged.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
SSDR	Store Segment Descriptor Registers	254 (1)

Summary: $C(AM.SDR) \Longrightarrow C(Y, Y+2, \ldots Y+30)_{0-71}$

where:

$C(SDR.ADDR) \Longrightarrow C(Y\ pair)_{0-23}$

$C(SDR.R1,R2,R3) \Longrightarrow C(Y\ pair)_{24-32}$

$0 \Longrightarrow C(Y\ pair)_{33-36}$

$C(SDR.BOUND) \Longrightarrow C(Y\ pair)_{37-50}$

$C(SDR.R,E,P,U,G) \Longrightarrow C(Y\ pair)_{51-56}$

$0 \Longrightarrow C(Y\ pair)_{57}$

$C(SDR.CL) \Longrightarrow C(Y\ pair)_{58-71}$

In their permanently assigned order, the contents of the segment descriptor word portion of the AM-SDW replace the contents of the 16 word pairs starting at location Y. The hardware shall assume $Y_{13-17} = 0000$ and addressing will be incremented accordingly. No check shall be made. The contents of the AM-SDW registers shall be unchanged. The associative memory is ignored (forced to a "no match") during the address generation phase.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
SPTR	Store Page Table Registers	154 (1)

Summary: $C(AM.PTR) \Longrightarrow C(Y, Y+1, \ldots Y+15)_{0-35}$

$C(PTR.ADDR) \Longrightarrow C(Y)_{0-17}$

$00 \ldots 00 \Longrightarrow C(Y)_{18-28}$

$C(PTR.M) \Longrightarrow C(Y)_{29}$

$00 \ldots 00 \Longrightarrow C(Y)_{30-35}$

In their permanently assigned order, the contents of the page table word portion of the AM-PTW replace the contents of the 16 words starting at location Y. The hardware shall assume $Y_{14-17} = 0000$ and addressing will be incremented accordingly. No check shall be made. The contents of the AM-PTW registers shall be unchanged. The associative memory is ignored (forced to a "no match") during the address generation phase.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
SSDP	Store Segment Descriptor Pointers	557 (0)

Summary: $C(AM.SDP) \Longrightarrow C(Y,Y+1 \ldots Y+15)_{0-35}$

where:

$C(SDP.POINTER) \Longrightarrow C(Y)_{0-14}$

$00 \ldots 00 \Longrightarrow C(Y)_{15-26}$

$C(SDP.F) \Longrightarrow C(Y)_{27}$

$00 \ldots 00 \Longrightarrow C(Y)_{28-31}$

$C(SDP.USE) \Longrightarrow C(Y)_{32-35}$

In their permanently assigned order, the contents of the POINTER, full/empty bit, and USE field of the AM-SDW replace the contents of the 16 word pairs starting at location Y. The hardware shall assume $Y_{14-17} = 0000$ and addressing will be incremented accordingly. No check shall be made. The contents of the AM-SDW registers shall be unchanged. The associative memory is ignored (forced to "no match") during the address generation phase.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Note: This and SPTP replaces 645 Store Associative Memory instruction.

Mnemonic	Name of Instruction	Op Code (octal)
SPTP	Store Page Table Pointers	557 (1)

Summary: $C(AM.PTP) \Longrightarrow C(Y, Y+1 \ldots Y+15)_{0-35}$

where:

$C(PTP.POINTER) \Longrightarrow C(Y)_{0-14}$

$C(PTP.PAGENO) \Longrightarrow C(Y)_{15-26}$

$C(PTP.F) \Longrightarrow C(Y)_{27}$

$00....00 \Longrightarrow C(Y)_{28-31}$

$C(PTP.USE) \Longrightarrow C(Y)_{32-35}$

In their permanently assigned order, the contents of the POINTER, PAGENO, full/empty bit, and USE field of the AM-PTW replace the contents of 16 words starting at location Y. The hardware shall assume $Y_{14-17} = 0000$ and addressing will be incremented accordingly. No check will be made. The contents of the AM-PTW registers shall be unchanged. The associative memory is ignored (forced to "no match") during the address generation phase.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
SRA	Store Ring Alarm	754 (1)

Summary: $00 \ldots 0 \Longrightarrow C(Y)_{0-32}$

$C(RALR)_{0-2} \Longrightarrow C(Y)_{33-35}$

The contents of the Ring Alarm Register are stored in location Y as shown.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
ADWPn	Add to Word Register of PRn	050 (0) 051 (0) 052 (0) 053 (0) 150 (0) 151 (0) 152 (0) 153 (0)

Summary: $C(Y)_{0-17} + C(PRn.ADR)_{0-17} \Longrightarrow C(PRn.ADR)_{0-17}$

$00 \ldots 00 \Longrightarrow C(PRn.ADR)_{18-23}$

The contents of memory location Y is added to the contents of the WORD NO field of Pointer Register n and the result replaces the contents of the WORDNO field. Zeros replace the contents of the BIT field of Pointer Register n. The rest of Pointer Register n remains unchanged.

Modifications: All except CI, SC, SCR, DL

Indicators Affected: None

Illegal Procedure Fault:

Modifications: CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
STAC	Store a Conditional	354 (0)

Summary: Test C(Y) then:

(1) if C(Y) = 0, C(A) $\Rightarrow$ C(Y), zero indicator set ON

(2) if C(Y) $\neq$ 0, zero indicator set OFF

If the initial C(Y) is nonzero then C(Y) are not changed by this instruction.

Modifications: All types except DU, DL, CI, SC, SCR

Indicators Affected:

Zero	If initial C(Y) = 0, then ON; otherwise OFF

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
STACQ	Store a Conditional-C(Y)=C(Q)	654 (0)

Summary: Test C(Y) then:

(1) if C(Y) = C(Q), C(A) = C(Y), zero indicator set ON

(2) if C(Y) ≠ C(Q), zero indicator set OFF

If the initial C(Y) is ≠ C(Q) then C(Y) are not changed by this instruction.

Modifications: All types except DU, DL, CI, SC, SCR

Indicators Affected:

Zero	If initial C(Y) = C(Q) then ON: otherwise OFF

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
STCD	Store Control Double	357 (0)

Summary: $C(PPR) \Longrightarrow C(Y\ pair)$

$00\ .\ .\ .\ 00 \Longrightarrow C(Y\ pair)_{0-2}$

$C(PPR.PSR) \Longrightarrow C(Y\ pair)_{3-17}$

$C(PPR.PRR) \Longrightarrow C(Y\ pair)_{18-20}$

$00\ .\ .\ .\ 00 \Longrightarrow C(Y\ pair)_{21-29}$

$43_8 \Longrightarrow C(Y\ pair)_{30-35}$

$C(PPR.IC)+2 \Longrightarrow C(Y\ pair)_{36-53}$

$00\ .\ .\ .\ 00 \Longrightarrow C(Y\ pair)_{54-71}$

The procedure segment and ring number in the word pair starting at location Y as shown. The hardware shall assume $Y_{17} = 0$. No check shall be made.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
SCU	Store Control Unit	657 (0)

$C(PPR,TPR....) \Longrightarrow C(Y, Y+2, Y+4, Y+6)_{0-17}$

Summary: $C(PPR.PRR) \Longrightarrow C(Y)_{0-2}$; $C(PPR.PSR) \Longrightarrow C(Y)_{3-17}$;
APU Status $\Longrightarrow C(Y)_{18-29}$; Directed Fault CNTR $\Longrightarrow C(Y)_{30-32}$;
Ring Alarm Register $\Longrightarrow C(Y)_{33-35}$; $C(PPR.IC) \Longrightarrow C(Y+1)_{0-17}$;
$C(IR) \Longrightarrow C(Y+1)_{18-35}$;
$C(TPR.TRR) \Longrightarrow C(Y+2)_{0-2}$; $C(TPR.TSR) \Longrightarrow C(Y+2)_{3-17}$;
TSR Status $\Longrightarrow C(Y+2)_{18-29}$; Delta $\Longrightarrow C(Y+2)_{30-35}$;
Fault Data $\Longrightarrow C(Y+3)_{0-35}$; $C(TPR.CA) \Longrightarrow C(Y+4)_{00-17}$;
CU Status $\Longrightarrow C(Y+4)_{18-29}$
TAG $\Longrightarrow C(Y+4)_{30-35}$; 00 . . . . 00 $\Longrightarrow C(Y+5)_{0-35}$;
Inst Working Reg or Even Inst $\Longrightarrow C(Y+6)_{0-35}$; Odd
Inst $\Longrightarrow C(Y+7)_{0-35}$

Information required to service a fault and then return to the operation in process at the time of the fault replaces the contents of 4-word pairs starting at location Y. The hardware shall assume $Y_{15-17} = 000$ and addressing will be incremented accordingly. No check shall be made.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode

Notes:

1. Information stored using the instruction SCU, shall only be valid if this is the first instruction executed after a fault.

2. Fault Data $C(Y+3)_{0-35}$

Bit	Fault	Mnemonic
0	Illegal Ring Order	ACV
1	Not in execute bracket	ACV
2	Execute bits is off	ACV
3	Not in read bracket	ACV
4	Read bit is off	ACV
5	Not in write bracket	ACV
6	Write bit is off	ACV
7	Not a gate	ACV
8	Not in call bracket	ACV
9	Outward call	ACV
10	Bad outward call	ACV
11	Inward return	ACV
12	Cross ring transfer	ACV
13	Ring alarm	ACV
14	Associative memory	ACV
15	Out of segment bounds	ACV
16	Processor parity error upper	FPAR
17	Processor parity error lower	FPAR
18	SC to Proc. Seq. Error #1	FONC
19	SC to Proc. Seq. Error #2	FONC
20	Illegal Action Lines	Illegal actions defined in Section 3.3.7 655 Proc EPS1
21	Illegal Action Lines	Illegal actions defined in Section 3.3.7 655 Proc EPS1
22	Illegal Action Lines	Illegal actions defined in Section 3.3.7 655 Proc EPS1
23	Illegal Action Lines	Illegal actions defined in Section 3.3.7 655 Proc EPS1
24	Illegal Action Channel	Illegal actions defined in Section 3.3.7 655 Proc EPS1
25	Illegal Action Channel	Illegal actions defined in Section 3.3.7 655 Proc EPS1
26	Illegal Action Channel	Illegal actions defined in Section 3.3.7 655 Proc EPS1
27	Connect Channel	CON
28	Connect Channel	CON
29	Connect Channel	CON
30	Fault Number/Address	
31	Fault Number/Address	
32	Fault Number/Address	
33	Fault Number/Address	
34		
35	Fault/Interrupt	

Group I

Bit	Fault	Mnemonic	
0	Illegal Segment Number	STR	Group II
1	Illegal Op Code	IPR	
2	Illegal Address and Modifier	IPR	
3	Illegal Slave Procedure	IPR	
4	All Other Illegal Procedure	IPR	
5	Nonexistent Address	STR	
6	Out of Bounds	STR	

3.4.2 Transfer of Control Instructions

Mnemonic	Name of Instruction	Op Code (octal)
TSPn	Transfer and Set PRn	270 (0) 271 (0) 272 (0) 273 (0) 670 (0) 671 (0) 672 (0) 673 (0)

Summary: $C(PPR) \Longrightarrow C(PRn)$

where:

$C(PPR.PRR) \Longrightarrow C(PRn.RNR)$

$C(PPR.PSR) \Longrightarrow C(PRn.SNR)$

$C(PPR.IC)+1 \Longrightarrow C(PRn.ADR)_{0-17}$

$0 \ldots\ldots 00 \Longrightarrow C(PRn.ADR)_{18-23}$

$CA \Longrightarrow C(PPR.IC)$

$C(TPR.TSR) \Longrightarrow C9PPR.PSR)$

The contents of the procedure pointer register replace the contents of Pointer Register n. The effective segment address and a segment number (see above) replace the instruction counter and segment number of the procedure pointer register.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR.

Note: This replaces 645 Transfer and Set Base n instructions.

Mnemonic	Name of Instruction	Op Code (octal)
RCU	Restore Control Unit	613 (0)

Summary: $C(Y,Y+2,Y+4,Y+6)_{0-71} \Longrightarrow C(PPR,TPR,...)$ where:

$C(Y)_{00-02} \Longrightarrow C(PPR.PRR)$; $C(Y)_{3-17} \Longrightarrow C(PPR.PSR)$;

$C(Y)_{18-20} \Longrightarrow$ APU Status; $C(Y)_{30-32} \Longrightarrow$ Directed Fault CNTR;

$C(Y+1)_{0-17} \Longrightarrow C(PPR.IC)$; $C(Y+1)_{18-35} \Longrightarrow C(IR)$;

$C(Y+2)_{00-02} \Longrightarrow C(TPR.TRR)$; $C(Y+2)_{3-17} \Longrightarrow C(TPR.TSR)$;

$C(Y+2)_{18-29} \Longrightarrow$ TSR Status; $C(Y+2)_{30-35} \Longrightarrow$ Delta;

$C(Y+4)_{18-29} \Longrightarrow$ CU Status; $C(Y+4)_{30-35} \Longrightarrow$ TAG;

$C(Y+6)_{0-35} \Longrightarrow$ Even Inst; $C(Y+7)_{0-35} \Longrightarrow$ Odd Inst.

The contents of 4-word pairs starting at location Y shall replace the contents of the PPR, TPR, etc. as shown above. The hardware shall assume $Y_{15-17} = 000$ and addressing will be incremented accordingly. No check shall be made. The contents of location Y through Y+7 shall be unchanged.

Modifications: All except DU, DL, CI, SC, SCR.

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in nonprivileged mode

Mnemonic	Name of Instruction	Op Code (octal)
RTCD	Return Double	610 (0)

Summary: $C(Y\ pair) \Longrightarrow C(PPR)$

$C(Y\ pair)_{3-17} \Longrightarrow C(PPR.PSR)$

Maximum $C(Y)_{18-20}$, $C(TPR.TRR)$, $C(SDW.R1_Y) \Longrightarrow C(PRR)$

$C(Y\ pair)_{36-53} \Longrightarrow C(PPR.IC)$

If TPR.TRR > PPR.PRR ∴ TPR.TRR→PR0-7.RNR

otherwise no change to RNR

The contents of the word pair starting at location Y replaces the contents of the procedure segment and ring number. The hardware shall assume $Y_{17} = 0$. No check shall be made. The contents of the word pair starting at location Y shall be unchanged.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Note:
- If an access violation occurs when fetching the SDW for location Y, the PPR. PSR and PPR.PRR are not altered.
- Refer to Section 2.5 for detailed information on executing this instruction in the Absolute Mode.

Mnemonic	Name of Instruction	Op Code (octal)
CALL	Call	713 (0)

Summary: $C(TPR) \Longrightarrow C(PPR,PR7)$

where: if $C(TPR.TRR) < C(PPR.PRR)$ $\therefore$ $C(DSBR.STACK)||C(TPR.TRR \Longrightarrow C(PR7.SNR)$

if $C(TPR.TRR) = C(PPR.PRR)$ $\therefore$ $C(PR6.SNR) \Longrightarrow C(PR7.SNR)$

$00 \ldots 00 \Longrightarrow C(PR7.ADR)_{0-17}$

$00 \ldots 00 \Longrightarrow C(PR7.ADR)_{18-23}$

$C(TPR.TRR) \Longrightarrow C(PPR.PRR)$

$C(TPR.TSR) \Longrightarrow C(PPR.PSR)$

$C(TPR.CA) \Longrightarrow C(PPR.IC)$

$C(TPR.TRR) \Longrightarrow C(PR7.RNR)$

The SEGNO field of PR7 is replaced according to the TRR and PRR fields (see above) and the effective ring number, segment number and segment address replace the corresponding fields of the processor pointer register.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Note:
- If C(TRR) > C(PRR) an access violation fault is initiated and the instruction is not executed.
- Refer to Section 2.5 for detailed information on executing this instruction in the Absolute Mode.

3.4.3 Special Instructions

Mnemonic	Name of Instruction	Op Code (octal)
CAMS	Clear Associative Memory Segmented	532 (0)

Summary: $0 \Longrightarrow C(SDP.F)$

$0000\text{-}1111 \Longrightarrow C(AM\text{-}SDW.USE)$

The full/empty bit of each segment descriptor word associative memory register is set to zero and the usage counters are initialized to their pre-assigned values of 0000 through 1111. The rest of the register shall be unchanged.

Modifications: None

Indicators Affected: None

Illegal Procedure Fault:

- Attempted execution in the unprivileged mode

Notes:
1. This and CAMP replace 645 Clear Associative Memory instruction.
2. The execution of this instruction will enable the associative memory, if previously disabled by an associative memory fault (See Section 3.7.2 Access Violations), if $C(CA)_{17}$ is equal to one.

Mnemonic	Name of Instruction	Op Code (octal)
CAMP	Clear Associative Memory Paged	532 (1)

Summary: 0 ⟹ C(PTP.F)

0000-1111 ⟹ C(AM-PTW.USE)

The full/empty bit of each page table word associative memory register is set to zero and the usage counters are initialized to their pre-assigned values of 0000 through 1111. The rest of the registers shall be unchanged.

Modifications: None

Indicators Affected: None

Illegal Procedure Fault:

- Attempted execution in the unprivileged mode.

Notes:

1. This and CAMS replace 645 Clear Associative Memory instruction.
2. The execution of this instruction will enable the associative memory, if previously disabled by an associative memory fault (See Section 3.7.2 Access Violations), if $C(CA)_{17}$ is equal to one.

Mnemonic	Name of Instruction	Op Code (octal)
RCCL	Read Calendar Clock	633 (0)

Summary: 00 .....00 $\Longrightarrow$ $C(AQ)_{0-19}$

C(Calendar Clock) $\Longrightarrow$ $C(AQ)_{20-71}$

CA_{0-2} specify which Processor Port is to be used.

The contents of the clock in one of the System Controllers replaces the contents of the AQ register as shown. The contents of the clock is unchanged. The three most significant bits of C(CA) specify which Processor Port (i.e., which System Controller) shall be used.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
RMCM	Read Memory Controller Mask Register	233 (0)

Summary:

$C(MR)_{0-15} \Longrightarrow C(AQ)_{0-15}$

$00....00 \Longrightarrow C(AQ)_{16-31}$

$C(MR)_{32-35} \Longrightarrow C(AQ)_{32-35}$

$C(MR)_{36-51} \Longrightarrow C(AQ)_{36-51}$

$00....00 \Longrightarrow C(AQ)_{52-67}$

$C(MR)_{68-71} \Longrightarrow C(AQ)_{68-71}$

The contents of the Mask Register in one of the System Controllers replaces the contents of the AQ register as shown. The contents of the Mask Register remain unchanged. The three most significant bits of the C(CA) specify which Processor Port (i.e., which System Controller) shall be used.

Modifications:

All except DU, DL, CI, SC, SCR.

Indicators Affected:

Zero	If C(AQ) = 0, then ON; otherwise OFF

Negative	If $C(AQ)_0$ = 1, then ON, otherwise OFF

Illegal Procedure Fault:

- Modifications DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode.

Mnemonic	Name of Instruction	Op Code (octal)
SMIC	Set Memory Controller Interrupt Cells	451 (0)

Summary: For i = 0, 1,...,15 and $C(A)_{35}$ = 0:

if $C(A)_i$ = 1, then set Interrupt Cell (i) ON

For i = 0, 1,....,15 and $C(A)_{35}$ = 1:

if $C(A)_i$ = 1, then set Interrupt Cell (16+i) ON

The C(A) is used to set selected Interrupt Cells in one of the System Controllers. The three most signifcant bits of the C(CA) specify which Processor Port (i.e., which System Controller) shall be used.

Modification:

All except DU, DL, CI, SC, SCR

Indicators:

None affected.

Illegal Procedure Fault:

- Modifications DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode.

Mnemonic	Name of Instruction	Op Code (octal)
SMCM	Memory Controller Mask Register	553 (0)

Summary: $C(AQ)_{0-15} \Longrightarrow C(MR)_{0-15}$

$C(AQ)_{32-35} \Longrightarrow C(MR)_{32-35}$

$C(AQ)_{36-51} \Longrightarrow C(MR)_{36-51}$

$C(AQ)_{68-71} \Longrightarrow C(AQ)_{68-71}$

The C(AQ) replaces the contents of the Mask Register in the selected System Controller. The C(AQ) remain unchanged. The three most significant bits of the C(CA) specify which Processor Port (i.e., which System Controller) shall be used.

Modifications:

All except DU, DL, CI, SC, SCR

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in the unprivileged mode.

Mnemonic	Name of Instruction	Op Code (octal)
RSW	Read Switches	231 (0)

Summary: This instruction shall cause the processor data/configuration/fault base and processor number switches to be read into the A-register. The particular switches to be read shall be determined by the $C(CA)_{16-17}$ of the instruction. The address assignments shall be as follows:

Address (octal)	Meaning
XXXXX0	36 data switches on the processor panel
XXXXX1	Configuration switches (PORTS A, B, C, D)
XXXXX2	Fault Base and processor number switches
XXXXX3	Configuration switches (PORTS E, F, G, H)

Modifications: All allowed

Indicators:

ZERO	If C(A) = 0, then ON, otherwise OFF

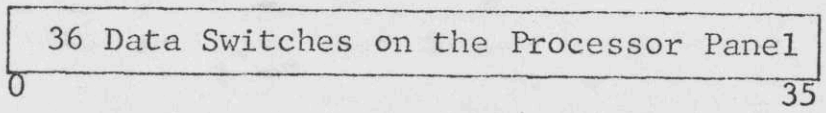
NEGATIVE	If C(A)0 = 1, then ON, otherwise OFF

Illegal Procedure Fault:

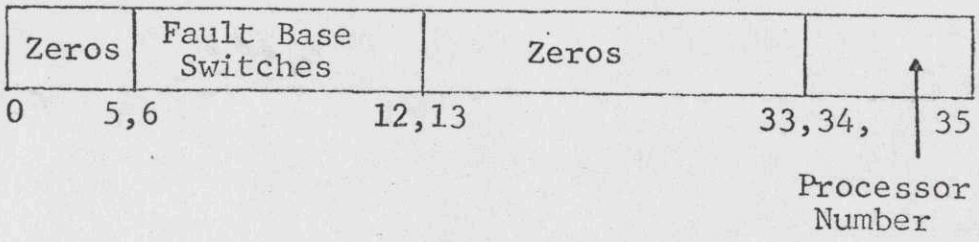
- Attempted execution in the unprivileged mode

NOTE: The format of the configuration switches shall be as follows:

For RSW address (octal) of XXXXX0:



For RSW address (octal) of XXXXX2:



The fields for the Fault Base and Processor number switches shall be coded as follows:

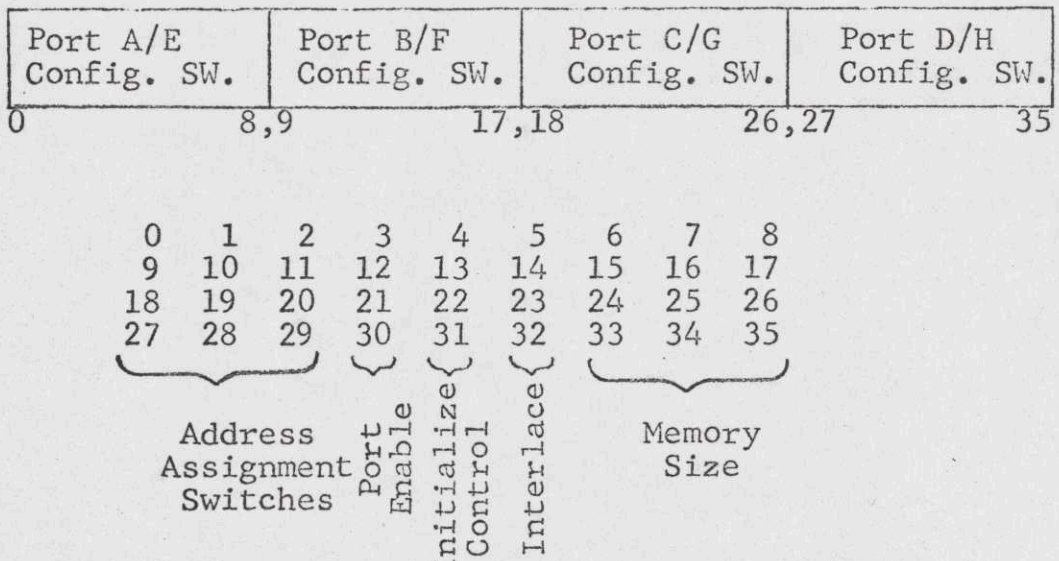
Field	Code	Meaning
Fault Base	0000000 to 1111111	Bits 6 to 12 of the fault vector addresses.

The fields for the Address Assignment Switches, Port Enable, Initialize Control, Interlace and Memory Size shall be coded as follows:

Field	Code	Meaning
Port Assignment	000 to 111	Reflects Port Address switches.
Port Enable	0	This port not configured into system.
	1	This port configured into system.
System Interlace Enable	0	Processor can't be initialized from system.
	1	Processor can be initialized from system.
Interlace Enable	0	Port not interlaced.
	1	Enables port to be interlaced with another port pair.
Memory Size	000	Not used
	001	32K
	010	64K

Field	Code	Meaning
Memory Size	011	128K
	100	256K
	101	512K
	110	1024K
	111	2048K
Processor Number	00	Processor Number 0
	01	Processor Number 1
	10	Processor Number 2
	11	Processor Number 3

For RSW address (octal) of XXXXX1/3:



Mnemonic	Name of Instruction	Op Code (octal)
MME2	Master Mode Entry 2	004 (0)

Summary: Causes a fault which obtains and executes in the Absolute Mode the two instructions stored at the memory locations 52+C and 53+C (octal); the constant C is set up on the Maintenance Panel FAULT VECTOR switches.

Execution of the MME2 instruction implies the following conditions:

1. During the execution of this MME2 instruction and the two instructions obtained, the processor is in the Absolute Mode independent of the value of its Absolute Mode indicator. The processor will stay in Absolute Mode if the Absolute Mode indicator is set ON after the execution of these three instructions.

2. The instruction from 52+C must not alter the memory location 53+C, and must not be an XED instruction.

3. If the instruction 52+C alters the contents of the instruction counter, then this transfer of control is effective immediately and the instruction from 53+C is not executed.

4. After the execution of the two instructions obtained from Y-pair, the next instruction to be executed is obtained from C(IC)+1. This is the instruction stored in memory right after this MME2 instruction unless the contents of the Instruction Counter have been changed by the execution of the two instructions obtained from 52+C and 53+C.

Modifications: Are without any effect on the operation.

Indicators Affected: None (see 3.6 for indicator reaction during execution of the vector pair).

Mnemonic	Name of Instruction	Op Code (octal)
MME3	Master Mode Entry 3	005 (0)

Summary: Causes a fault which obtains and executes in the Absolute Mode the two instructions stored at the memory location 54+C and 55+C (decimal); the constant C is set up on the Maintenance Panel FAULT VECTOR switches.

Execution of the MME3 instruction implies the following conditions:

1. During the execution of this MME3 instruction and the two instructions obtained, the processor is in the Absolute Mode independent of the value of its Absolute Mode indicator. The processor will stay in Absolute Mode if the Absolute Mode indicator is set ON after the execution of these three instructions.

2. The instruction from 54+C must not alter the memory location 55+C and must not be an XED instruction.

3. If the instruction from 54+C alters the contents of the Instruction Counter, then this transfer of control is effective immediately and the instruction from 55+C is not executed.

4. After the execution of the two instructions obtained from Y-pair, the next instruction to be executed is obtained from C(IC)+1. This is the instruction stored in memory right after this MME3 instruction unless the contents of the Instruction Counter have been changed by the execution of the two instructions obtained from 54+C and 55+C.

Modifications: Are without any effect on the preparation.

Indicators Affected: None (see 3.6 for indicator reaction during execution of the vector pair).

Mnemonic	Name of Instruction	Op Code (octal)
MME4	Master Mode Entry 4	007 (0)

Summary: Causes a fault which obtains and executes in the Absolute Mode the two instructions stored at the memory locations 56+C and 57+C (decimal); the constant C is set up on the Maintenance Panel FAULT VECTOR switches.

Execution of the MME4 instruction implies the following conditions:

1. During the execution of this MME4 instruction and the two instructions obtained, the processor is in the Absolute Mode independent of the value of its Absolute Mode indicator. The processor will stay in Absolute Mode if the Absolute Mode indicator is set ON after the execution of these three instructions.

2. The instruction from 56+C must not alter the memory location 57+C, and must not be an XED instruction.

3. If the instruction from 56+C alters the contents of the Instruction Counter, then this transfer of control is effective immediately and the instruction from 57+C is not executed.

4. After the execution of the two instructions obtained from Y-pair, the next instruction to be executed is obtained from C(IC)+1. This is the instruction stored in memory right after this MME4 instruction unless the contents of the Instruction Counter have been changed by the execution of the two instructions obtained from 56+C and 57+C.

Modifications: Are without any effect on the operation.

Indicators Affected: None (see 3.6 for indicator reaction during execution of the vector pair).

3.4.4 Modified H6000 Instructions

Mnemonic	Name of Instruction	Op Code (octal)
CIOC	Connect I/O Channel	015 (0)

Summary: C(Y) are transferred from the memory module via the memory module port that is specified by C(Y).

1. The absolute address Y is used to access a memory location as usual. However, the memory module does not transmit the contents of this location to the processor that submitted the absolute address; it uses $C(Y)_{33...35}$ to select one of its eight ports, sends a connect pulse to the unit on this port, and then transmits C(Y) on the data lines to this unit.

2. This instruction can be used in the privileged mode only.

3. A CON command is sent out that processor port selected by the 24 bit absolute address formed by this instruction.

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

- Modifications: DU, DL, CI, SC, SCR
- Attempted execution in unprivileged or BAR modes.

Mnemonic	Name of Instruction	Op Code (octal)
RET	Return	630 (0)

Summary: $C(Y)_{0-17} \Longrightarrow C(IC)$; $C(Y)_{18-31} \Longrightarrow C(IR)$; $C(Y)_{32-35}$ are not used.

The contents of the location specified by Y replace the contents of the Instruction Counter and Indicator Register. A possible change in status of the Absolute mode indicator takes place as the last part of the instruction execution. The Tally Runout indicator reflects the state of $C(Y)_{25}$ prior to address modification and is unaffected by any subsequent modification performed on the RET instruction. The relationship between $C(Y)_{18-31}$ and the indicators are as follows:

Bit Position C(Y)	Indicator
18	Zero
19	Negative
20	Carry
21	Overflow
22	Exponent Overflow
23	Exponent Underflow
24	Overflow Mask
25	Tally Runout
26	Parity Error
27	Parity Mask
28	BAR Mode
29	Truncation
30	Multiword Instruction Int.
31	Absolute Mode

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected:

Parity Mask	If corresponding bit in C(y) is 1, and processor is in the absolute or privileged mode, then ON; otherwise OFF. Indicator is not affected in the nonprivileged or BAR modes
BAR Mode	If the corresponding bit in C(y) is 1, then NO CHANGE. If the corresponding bit in C(y) is 0, then OFF.
Absolute Mode	If the corresponding bit in C(y) is 1, then NO CHANGE. If the corresponding bit in C(y) is 0, then OFF.
All Other Indicators	If corresponding bit in C(y) is 1, then ON; otherwise OFF.

Mnemonic	Name of Instruction	Op Code (octal)
STC1	Store Instruction Counter plus 1	554 (0)

Summary: $C(ICO) + 0...01 = C(Y)_{0-17}$

$C(IR) \Longrightarrow (Y)_{18-31}$; $00...0 = C(Y)_{32-35}$

The contents of the Instruction Counter and the Indicator Register are stored in $C(Y)_{0-17}$ and $C(Y)_{18-31}$, respectively after modification. The $C(Y)_{25}$ reflects the state of the Tally Runout indicator prior to modification. The relationship between the $C(Y)_{18-31}$ and the indicators are as follows:

Bit Position C(Y)	Indicators
18	Zero
19	Negative
20	Carry
21	Overflow
22	Exponent Overflow
23	Exponent Underflow
24	Overflow Mask
25	Tally Runout
26	Parity Error
27	Parity Mask
28	BAR Mode
29	Truncation
30	Multiword Inst. Int.
31	Absolute Mode

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
SREG	Store Registers	753 (0)

Summary: C(X0,X1,...X7,A,Q,E,TR) = C(Y,Y+1,...Y+7)
where Y must be an 0 module 8 address.*

The contents of the index (X0-X7), A-,Q-,E- and TR-registers are stored in successive locations beginning at Y and ending at Y+7 in the following format:

$C(X0) \Longrightarrow C(Y)_{0-17}$

$C(X1) \Longrightarrow C(Y)_{18-35}$

$C(X2) \Longrightarrow C(Y+1)_{0-17}$

$C(X3) \Longrightarrow C(Y+1)_{18-35}$

$C(X4) \Longrightarrow C(Y+2)_{0-17}$

$C(X5) \Longrightarrow C(Y+2)_{18-35}$

$C(X6) \Longrightarrow C(Y+3)_{0-17}$

$C(X7) \Longrightarrow C(Y+3)_{18-35}$

$C(A) \Longrightarrow C(Y+4)_{0-35}$

$C(Q) \Longrightarrow C(Y+5)_{0-35}$

$C(E) \Longrightarrow C(Y+6)_{0-7}$;
$00...0 \Longrightarrow C(Y+6)_{18-35}$

$C(TR) \Longrightarrow C(Y+7)_{0-26}$,
$00...0 \Longrightarrow C(Y+7)_{24-32}$

$C(RALR) \Longrightarrow C(Y+7)_{33-35}$

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

*The hardware shall assume $Y_{15-17} = 000$ and addressing will be incremented accordingly. No check will be made.

Mnemonic	Name of Instruction	Op Code (octal)
LDI	Load Indicator Register	634 (0)

Summary: $C(Y)_{18-30} = C(IR)$

The relationship between $C(Y)_{18-30}$ and the indicators is as follows:

Bit Position C(Y)	Indicators
18	Zero
19	Negative
20	Carry
21	Overflow
22	Exponent Overflow
23	Exponent Underflow
24	Overflow Mask
25	Tally Runout
26	Parity Error
27	Parity Mask
29	Truncation
30	Multiword Instruction Int.

Modifcations: All except CI, SC, SCR.

The tally runout indicator will reflect $C(Y)_{25}$ regardless of what address modification is performed on the LDI instruction for tally operations.

Indicators Affected:

Parity Mask	If corresponding bit in C(Y) is 1, and processor is in the Absolute or Privileged Mode, then ON; otherwise OFF. Indicator is NOT AFFECTED in the Nonprivileged or BAR Modes.
BAR Mode	Not affected.
Absolute Mode	Not affected.
All Other Indicators	If corresponding bit in C(Y) is ONE, then ON; otherwise OFF.

Illegal Procedure Fault:

Modifications: SC, SCR, CI

Mnemonic	Name of Instruction	Op Code (octal)
STI	Store Indicator Register	754 (0)

Summary: $C(IR) = C(Y)_{18-31}$

The contents of the Indicator Register are stored in $C(Y)_{18-31}$ after modification, but the $C(Y)_{25}$ reflects the state of the Tally Runout indicator prior to modification. The relationship between $C(Y)_{18-31}$ and the indicators are as follows:

Bit Position C(Y)	Indicator
18	Zero
19	Negative
20	Carry
21	Overflow
22	Exponent Overflow
23	Exponent Underflow
24	Overflow Mask
25	Tally Runout
26	Parity Error
27	Parity Mask
28	BAR Mode
29	Truncation
30	Multiword Instruction Int.
31	Absolute Mode

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TRA	Transfer Unconditionally	710 (0)

Summary: EA $\Longrightarrow$ C(IC), C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TSXn	Transfer and Set Index Register n(n=0,1...,7)	70n (0)

Summary: C(IC) + 0...01 ⟹ C(Xn) EA ⟹ C(IC),

C(TSR) ⟹ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TBR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modification: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TPL	Transfer on Plus	605 (0)

Summary: If Negative Indicator OFF, then EA $\Longrightarrow$ C(IC), C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TRC	Transfer on Carry	603 (0)

Summary: If Carry Indicator ON, then EA $\Longrightarrow$ C(IC),
C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, CL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TNC	Transfer on No Carry	602 (0)

Summary: If Carry Indicator OFF, then EA $\Longrightarrow$ C(IC), C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TOV	Transfer on Overflow	617 (0)

Summary: If Overflow Indicator ON, then EA $\Longrightarrow$ C(IC), C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected:

Overflow	Set OFF

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TEU	Transfer on Exponent Underflow	615 (0)

Summary: If Exponent Underflow indicator ON then EA $\Longrightarrow$ C(IC); C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected:

Exponent Underflow	Set OFF

Illegal Procedure Fault:

Modifications: DU, DL, SC, CI, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TEO	Transfer on Exponent Overflow	614 (0)

Summary: If Exponent Overflow indicator ON; then EA $\Longrightarrow$ C(IC); C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SCR

Indicators Affected:

Exponent Overflow	Set OFF

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TMI	Transfer on Minus	604 (0)

Summary: If Negative Indicator ON, then EA $\Longrightarrow$ C(IC); C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TRTN	Transfer on Truncation Indicator On	600 (1)

Summary: If truncation indicator ON, then EA ⟹ C(IC); C(TSR) ⟹ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected:

Truncation	Set OFF

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TPNZ	Transfer on Plus and Nonzero	602 (1)

Summary: If Negative and Zero Indicators are OFF, then EA $\Longrightarrow$ C(IC); C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TNZ	Transfer ON not Zero	601 (0)

Summary: If Zero Indicator OFF, then EA $\Longrightarrow$ C(IC);
C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TZE	Transfer On Zero	600 (0)

Summary: If Zero Indicator ON, then EA $\Longrightarrow$ C(IC);
C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TTF	Transfer on Tally Runout Indicator Off	607 (0)

Summary: If Tally Runout Indicator OFF, then
EA $\Longrightarrow$ C(IC); C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TTN	Transfer on Tally Runout Indicator ON	603 (1)

Summary: If Tally Runout Indicator ON, then EA $\Longrightarrow$ C(IC);
C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TRTF	Transfer on Truncation Indicator OFF	601 (1)

Summary: If truncation indicator OFF, then EA $\Longrightarrow$ C(IC);
C(TSR) $\Longrightarrow$ C(PSR)

The new effective address replaces the C(IC), and the new pointer in TSR formed during the appending process for the transfer address replaces C(PSR).

Modifications: All except DU, DL, CI, SC, SCR.

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, CI, SC, SCR

Mnemonic	Name of Instruction	Op Code (octal)
TSS	Transfer and Set Slave	715 (0)

Summary: EA $\Longrightarrow$ C(IC); C(TSR) $\Longrightarrow$ C(PSR)

The new Effective Address replaces the C(IC), and the new Pointer in TSR formed during the Appending process for the transfer address replaces C(PSR).

If this instruction is executed while in the MULTICS operational mode, then the Absolute indicator is turned off and the BAR Mode indicator will reset ($\overline{\text{BAR}}$ Mode). The BAR is then used in the final address preparation of the transfer.

If this instruction is executed with the BAR Mode indicator reset (in $\overline{\text{BAR}}$ Mode) then it functions as a TRA instruction.

Modifications: All except DU, DL, CI, SC, SCR.

Indicators Affected: None

Illegal Procedure Fault:

Modifications: DU, DL, SC, CI, SCR

NOTE: Refer to Section 2.5 for detailed information on executing this instruction in the Absolute Mode.

3.5 INDICATOR REGISTER

The Indicator Register shall be the same as the H6000 with the addition that bit 31 of the Indicator Register shall be used as the Absolute Mode Indicator. The indicator shall be set and reset as described in Section 2.2.

3.6 645F PROCESSOR FAULTS AND INTERRUPTS

3.6.1 GECOS Mode

The 645F Processor when operating in the GECOS Mode shall recognize 16 faults. The mnemonics, trapping procedure, and categories for fault and interrupt conditions shall be identical with those described in the 655 EIS Processor EPS-1.

3.6.2 MULTICS Mode

The 645F Processor when operating in the MULTICS Mode shall recognize an extended set of 27 faults as indicated in Table 3.7. Faults 1 through 16 in the 645F Processor shall be assigned identically for both MULTICS and GECOS Modes. Sampling and processing shall be as described in the 655 EIS Processor EPS-1.

3.7 645F MULTICS PROCESSOR FAULTS

3.7.1 Fault Categories

There are four general fault cagegories. Certain faults may fall under more than one category. The categories and the faults, which may belong to them at one time or another, are as follows:

1. Instruction or program generated -

   Master Mode Entry 1-4

   Derail

   Fault Tag 1-3

   Connect

   Illegal Procedure

   Store Fault

Table 3.7 - 645F Faults

Mnemonic	Name	Fault Code	Priority	Group
SUF	Start-up	30	1	I
EXF	Execute	36	2	I
TRB	*Trouble	76	3	II
FONC	Op Not Complete	26	4	II
FDIV	Divide Check	34	6	III
FOFL	Overflow	32	7	III
LUF	Lock-up	16	5	IV
FPAR	Parity	22	8	IV
FCMD	Command	12	9	IV
STR	Store	02	10	IV
MME	Master Mode Entry 1	04	11	V
MME2	*Master Mode Entry 2	52	12	V
MME3	*Master Mode Entry 3	54	13	V
MME4	*Master Mode Entry 4	56	14	V
DRL	Derail	14	15	V
IPR	Illegal Procedure	24	16	V
FTG_1	Fault Tag 1	06	17	V
FTG_2	*Fault Tag 2	60	18	V
FTG_3	*Fault Tag 3	62	19	V
DFT_0	*Directed Fault 0	40	20	VI
DFT_1	*Directed Fault 1	42	21	VI
DFT_2	*Directed Fault 2	44	22	VI
DFT_3	*Directed Fault 3	46	23	VI
ACV^3	*Access Violation	50	24	VI
CON	Connect	20	25	VII
TROF	Timer Runout	10	26	VII
SDF	Shutdown	00	27	VII

*These faults are enabled in the MULTICS Mode.

Timer Runout
Lockup
Overflow
Divide Check
Command Fault

2. Hardware generated -

Command Fault
Parity
Operation Not Completed
Trouble
Lockup

3. Manually generated -

Startup
Shutdown
Execute

4. Operating system generated -

Directed Faults 0-3
Access Violation
Trouble
Any instruction or program generated fault caused by the operating system.

3.7.2 Fault Descriptions

1. Startup

A power turn-on has occurred. When the POWER ON button is depressed the processor is first initialized and then the startup fault is recognized.

2. Execute

   a. The EXECUTE pushbutton on the processor maintenance panel has been depressed.

   b. An external frequency (for example, a scope gate) has been substituted for the EXECUTE pushbutton.

   The above two are dependent on other maintenance panel switch positions. Also, certain switch positions will affect execution of the instruction set up on the instruction switches, rather than the fault vector in storage. In either case, the processor is first initialized and then the execute fault is recognized.

3. Trouble

   The trouble fault is defined as the occurrence of a fault during the execution of a vector pair forced by a fault or external interrupt, or only during the first instruction of the vector pair if it is a transfer of control. Such faults may be hardware generated (for example, Operation Not Complete or Parity), or operating system generated (for example, a page containing the indirect address of an instruction is missing).

4. Operation Not Completed

   a. The processor has addressed a system controller to which it is not attached.

   b. The addressed system controller failed to acknowledge the processor.

   c. The processor has not generated a storage access request or a direct operand within 1 to 2 milliseconds and is not in the DIS state.

   d. A processor port received a data strobe without a preceding acknowledgement from the system controller that it had received the interrupt.

   e. A processor port received a data strobe before the data previously sent to it was unloaded.

5. Lock Up

The program is in a program lock-up which inhibits sampling for an external (whether present or not) interrupt or interrupt type fault (Group VII). The lock-up time shall be specified by the Lock-Up Fault Register, which may be set via program to 2, 4, 8 or 16 milliseconds.

6. Divide Check

A divide check fault occurs when the actual division cannot be carried out for one of the reasons specified with each divide instruction.

7. Overflow

An arithmetic overflow, exponent overflow, or exponent underflow has been generated. The generation of this fault is inhibited with the Overflow Mask indicator in the mask state. Subsequent clearing of the Overflow Mask indicator to the unmasked state will not generate this fault from previously set Overflow indicators. The Overflow Fault Mask state does not affect the setting, testing or storing of indicators. The determination of the specific overflow condition is by indicator testing by the control program.

8. Parity

This fault is generated when a parity error exists in a word which is read from a core location. If the parity error occurs on an instruction or an indirect word, the word will not be used any further and the fault routine will be entered. If the parity error occurs on an operand, the processor operation is aborted.

The generation of this fault is inhibited when the Parity Mask indicator is in the mask state. Subsequent clearing of the Parity Mask indicator to the unmasked state will not generate this fault from a previously set Parity Error indicator. The Parity Mask does not affect the setting, or storing of the Parity indicator.

If the parity error occurs on a SDW or PTW, the word will not be used and the fault routine will be entered regardless of the state of the Parity Mask indicator.

9. Command Fault

An attempt by the processor to execute the SMCM or SSCR instruction or the XEC command without an interrupt mask register assigned to the system controller port to which the processor is connected shall generate a FCMD fault.

A conditional illegal action may be set to cause the FCMD fault using the switches on the system controller maintenance panel.

Also, if processor issues a Connect to a channel that is masked off (by program or switch) a FCMD fault will be issued.

10. Access Violation

The Access Violation (ACV) is appending unit oriented. There are multiple violations the appending will detect causing an ACV fault. The different violations are:

a. Bad read bracket
b. Bad write bracket
c. Bad execute bracket
d. No read permission
e. No write permission
f. No execute permission
g. Invalid ring crossing
h. Call limiter fault
i. Outward call
j. Inward return
k. Ring alarm
l. Associative Memory
m. Out of segment bounds

A fault code register will be used to differentiate among the above access violations. A unique code will be set in the fault code register at the time of the fault. See Section 3.4.1.

11. Illegal Procedure

An illegal procedure fault may occur due to the following illegal actions:

a. Illegal Operation code

b. An illegal address (for instructions using the address field to specify a register)

c. An illegal modifier or modifier sequence

d. An illegal instruction sequence

e. An illegal instruction in nonprivileged mode.

CIOC	SCPR
SCU	RMCM
RCU	SMCM
LRA	DIS
LDT	SMIC
LDBR	ABSA
SDBR	RSCR
LCPR	

A fault code register will be used to differentiate between different illegal procedures (see Section 3.4.1).

12-15. Master Mode Entry 1-4

The instruction MME1-4 has been encountered.

16. Derail

The instruction DRL has been encountered.

17-19. Fault Tag 1-3

Fault Tag 1-3 code has been encountered in the tally designator field of an IT address modifier. The indirect cycle will not be made, nor will the operation be completed.

20-23. Directed Faults 0-3

A directed fault is inserted in the SDW or PTW by the operating system to prevent access to the segment or page.

Action by the operating system may be required before the sequence is resumed. Bits 34 and 35 of the SDW or PTW indicate the number of the directed fault, while bit 33=0 indicates a directed fault is present.

24. Connect

The processor has received a connect pulse from another active device through the system controller. This event is to be distinguished from a CIOC (connect) instruction encountered in the program sequence.

25. Timer Runout

This fault is generated when the contents of the timer register reaches zero. If the processor is in privileged mode, recognition of this fault will be delayed until the processor returns to the unprivileged mode; this delay does not inhibit the counting in the timer register. Since this condition is saved by the hardware, the operating system must exercise care not to transfer it from one user to another.

26. Shutdown

Power will be turned off in approximately one millisecond.

27. Store

The processor attempted to select a disabled port, an out-of-bounds address was generated in the BAR Mode or an attempt was made to access a store unit which was not ready.

3.7.3 Fault Priority

Two or more faults may exist in the processor concurrently due to the "look ahead" feature (that is, overlapping of instruction execution and address preparation). The fault which occurs first through normal program sequence will be serviced. The other fault flags are not safe-stored and later recognition is contingent on recursion. The only exceptions are Group VII faults which are saved by the heardware for eventual recognition.

The 27 faults are classified into seven groups (I-VII) to establish priority when faults occur simultaneously. Group I has the highest priority; Group VII has the lowest. The only exception is a Group III fault (Overflow or Divide Check) caused by an operand having a parity error.

In this case the parity fault (Group IV) will take precedence. In the case of simultaneous faults within Group VII, Shutdown has the highest priority, Timer Runout is next, and Connect is the lowest.

No simultaneity of faults within Groups (III, IV, V, VI) is expected, since they are mutually exclusive.

3.7.4 Fault Recognition

Faults in Group I cause the processor to initialize and enter the fault routine unconditionally. Operation Not Completed causes the processor to fault unconditionally.

When instructions which may cause Group III faults are executed, the processor is prevented from going ahead with any irrevocable action. Therefore, if an overflow or a divide check is generated, the processor immediately halts program execution.

Faults in Group IV cause the operations in the processor to abort conditionally upon the completion of the operations presently being executed.

Faults in Group V or VI are discovered during the address processing cycle of an instruction. Since this will happen most often in the look ahead mode, the processor will abort conditionally upon completion of all pending operations. If the

execution of any pending instructions causes a fault, it will override the Group V or VI faults.

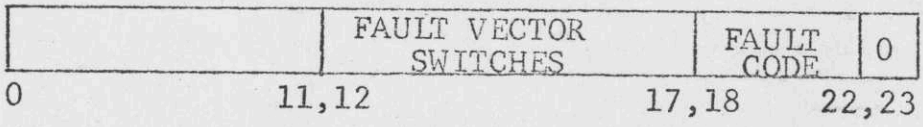
Faults in Groups III-VI must wait for the system controller to acknowledge the last access request before initiating the abort routine.

Faults in Group VII are recognized under similar conditions to external interrupts. The sampling of any one of these faults may be inhibited in the Privileged* mode by the interrupt inhibit bit (28).

3.7.5 Fault Vector Address

Each fault has a corresponding pair of instructions, known as the fault vector pair, to which it is trapped by the hardware. The base address of the fault vector block (as a 0 mod 64 number) is set on the maintenance panel or reconfiguration console via the FAULT VECTOR BASE ADDRESS switches.

To generate the fault vector address, the fault code is extended by a trailing zero (doubled) and concatenated to the value of the FAULT VECTOR switches. The resultant 24-bit number is the core location (absolute address) of the desired fault vector pair.



3.7.6 Fault Sequence

Following the detection of a fault condition, the control unit will determine the proper time to initiate the fault mode according to the type of fault. At that time the control unit aborts the processor. At the end of the abort cycle, the address of the fault vector pair is prepared.

* Implies that the processor procedure segment is privileged and the procedure ring (level of privilege) is equal to "0".

The XED op code and the fault vector pair's address are forced into the instruction register. Since the brute forced XED does not specify any address modification (t_m=R, t_d=N), the address formed by concatenating the fault code and the FAULT VECTOR switches becomes the core location. The processor will then issue a store access request for the vector pair. When the processor enters the XED instruction, it sets a Temporary Absolute Mode flag not affecting the Absolute indicator. Also, the mode of execution of the XED and the vector pair is set to privileged.

If the attempt to execute the XED results in a Parity or a Group II fault, the process will instead process a trouble fault. The safe-stored information remains the same, but a new XED instruction is brute-forced with an address corresponding to the trouble fault.

If the execution of the XED is successful, the processor control unit will begin address preparation for the even instruction, which may be a SCU.

If a Directed fault, Out-of-Bounds, or Access Violation fault is encountered during any address preparation cycle for the SCU operand or indirect word; or if a parity error is detected following the fetch of an indirect word, the Trouble fault routine will be entered as discussed previously.

After the first instruction has been completed successfully and did not result in transfer of control, the processor will begin address preparation for the second instruction. If either of the two instructions results in an actual transfer of control other than TSS, or a transfer with bit 29 = 1, or ITS indirection, then the Absolute indicator will be "on." The mode of execution will remain privileged unconditionally for as long as the Absolute indicator is "on." If the Absolute indicator is "off", the mode of execution will be set according to Privileged Mode indicator of the current procedure.

The two instructions of the fault vector must adhere to the programming rules of normal XED.

If neither of the XED'ed instructions causes a transfer of control the hardware will execute the instruction next in normal sequence (ICT+1) to the instruction which caused the fault directly, or indirectly will be executed.

3.7.7 Status Safestore and Restore

Pertinent information is registered by the processor in anticipation of a fault. This information is intended to be used by the hardware to restart the processor if and when necessary, and by the software to define the fault and course of action more specifically The safestore data which is registered contains the contents of various registers, status flags and a computed address for a total of four double words.

3.7.7.1 Group I Faults

Group I faults are brute-forced and the registered data is undefined.

3.7.7.2 Group II Faults

Group II faults are brute-forced, but in the case of an Op Not Complete, the ICT points to an uncompleted instruction. The safe-stored data of the Trouble fault is that of the original fault.

3.7.7.3 Group III Faults

Group III faults are detected during the execution of the instruction. The only valid safe-stored information is the Instruction Counter and Procedure Segment Register which point to the faulting instruction.

3.7.7.4 Group IV Faults

Group IV faults are detected after the processor control unit address preparation cycle has been completed, allowing the control unit to step into the next address preparation cycle with the

exception of the STR (00B+NEA) which is processor generated. The remaining Group IV faults are initiated by the system controller illegal action lines. Excluding the STR (00B+NEA), the only valid safe-stored information is the ICT and PSR which point to the faulting instruction. The SCU data safestored for the STR (00B+NEA) in conjunction with a RCU instruction may be used to restart the instruction from the point of interruption.

3.7.7.5 Group V Faults

Included in this group are the faults which are program generated. In this group are the MME's, Flt Tag's, IPR's and the DRL instruction. The detection of a Group V fault results in only the ICT and PSR as valid information pointing to the faulting instruction.

3.7.7.6 Group VI Faults

The Group VI fault is detected during the appending unit address preparation cycle. The control unit information registered contains valid information. The use of the SCU and RCU instruction may allow a faulting non-EIS type instruction to be restarted from the point of interruption.

The EIS instructions fall into two categories, those instructions whose arguments or final result is limited to 63 digits and the instructions which may exceed the 63 digit limitation. Instructions which fall into the latter category are defined as being interruptable and are defined in Section 3.6 of the 655 EIS Processor EPS-1.

An EIS instruction which is noninterruptable (results limited to 63 digits) which encounters a Group VI fault during execution will be restarted from the beginning, not the point of interruption. Instructions falling into the interruptable category which encounter a Group VI fault during execution will be restarted from the point of interruption. The procedure for restarting will be the same as the Multiword Instruction Interrupt Procedure defined in Section 3.6 of the 655 EIS Processor EPS-1.

3.7.7.7 Group VII Faults

Group VII faults fall into the same cagegory as external interrupts. The sampling of interrupts and Group VII faults is at the convenience of the hardware. The only vlid safe-stored information is the IC and PSR which point to the last successfully completed instructions. Exceptions to the above rule are the EIS MOVE instructions which are selectively interrupted, and through the use of special EIS instructions (LPL, SPL), pointer and length information is saved allowing the instructions to be restarted at the point of interruption.

3.7.8 645F Processor Interrupts

The processing of interrupts and external faults in the 645F will be identical to their operation in the H6000 series.

3.7.8.1 External Interrupts

When one, or more, of the unmasked execute interrupt cells in the system controller (SC) is set to a "1," the system controller then activates the execute interrupt present line to the control processor. Each execute interrupt cell in system controller has an instruction pair associated with it. The absolute address formed by the processor is a function of the system controller cell number.

The first instruction of the interrupt vector pair is expected to store the processor status so that a return can be made to the point of interruption. The second instruction is expected to cause a transfer to a service routine. If neither of the instructions in the vector pair is an instruction which will cause a transfer, the processor will execute the next instruction (ICT+1).

The processor samples the eight interrupt present lines from the system controller to which it is connected and strobes the information into its execute interrupt present register (XIPR). If an interrupt is present, generally the processor

upon completion of the execution of the odd instruction, issues an execute interrupt command (XEC) to the system controller corresponding to the highest priority XIPR which is set.

The system controller will respond with an encoded number (5 bits) on data lines 12-16. The processor receives the system controller cell number as part of the address of the vector pairs. This address is used with a forced XED instruction to retrieve the instruction pair from storage.

3.7.8.2 External Faults

Three events are treated as external faults by the processor and are sampled at the same time as interrupts. These are Connect, Timer Runout and Power Shutdown. These faults are stored in a register asynchronously as they occur. The register can be considered an extension of the XIPR and having a higher priority.

3.7.8.3 Interrupt Inhibit Bit

Bit 28 of the instruction word is defined as the interrupt inhibit bit. When the processor is executing the Ring "0" and Privileged Mode and the current instruction has a "1" in position 28, the recognition of external interrupts and faults are inhibited. Sampling of interrupts will be resumed upon completion of the execution of an instruction in an odd location or either Ring "0" or the Privileged Mode is exited.

3.7.8.4 Lock-Up Fault

The Lock-Up fault will occur when the processor fails to sample interrupts for a period designated by the Lock-Up Fault Mode Register. The Lock-Up Mode Register may be set to 2, 4, 8, or 16 millisecond time out sequences.

The coding of a single instruction (odd location) with a zero in bit 28 will cause the processor to sample interrupts and reinitiate the lock-up time out period.

The lock-up is deactivated in the DIS state.

3.7.8.5 DIS

In the DIS, or initialized state, no operation takes place, and the processor will wait for an external fault or interrupt. If a DIS is executed in something other than the absolute or privileged Ring "0" mode an illegal procedure (illegal in slave) fault will occur.

3.7.8.6 Interrupt Sampling

In the 645F Processor the interrupt sampling period takes place when the execution of an odd instruction has been completed. Also, the processor will not recognize interrupts under the following conditions:

1. All Repeat instructions and the repeated instructions.

2. Unconditional Transfer instructions or conditional transfers which have been met.

3. The processor is executing in Ring "0" in the Privileged Mode and bit 28 has been encountered in the instruction.

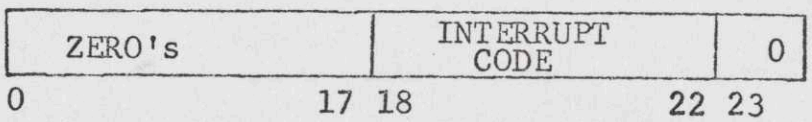
4. EIS instructions which are not defined as long MOVE's as defined in the 655 EIS Processor EPS-1.

5. The XEC and XED instructions and the executed instruction. Also, interrupts are inhibited during the execution of the interrupt or fault vector pair.

6. Multioperand type instructions (for example, LREG, SREG, LPRI, SPRI).

3.7.8.7 Interrupt Vector Address

The absolute address formed by the processor is a function of the system controller cell number. The processor upon receiving the interrupt cell number from the system controller will force an XED into the instruction register. The least significant bit of the address of the XED is always "0". The next five bits are the interrupt cell number received from the system controller as a result of the XEC command. It should be noted that the interrupt vectors for the 645F will always reside in core locations $0\text{-}64_8$. The absolute 24 bit address is the core location of the desired interrupt vector pair.



3.8 CONFIGURATION CONTROL

The configuration of the processor shall be controlled from a panel that contains the following controls:

- Address Assignment Switches
- Interlace Switches
- Port Enable Switches
- Initialize Enable Switches
- Size Switches
- Processor Number Switches
- Fault Base Switches

3.8.1 Address Assignment Switches

The Address Assignment switches shall consist of three toggle switches for each port. They shall indicate the memory block assigned to that port.

3.8.2 Interlace Switches

There shall be one 3-position switch for each pair of ports. The following pairs of ports may be interlaced.

Ports A and B

Ports C and D

Ports E and F

Ports G and H

The interlace mode may either be two (address bit 22) or four (address bit 21) word blocks.

3.8.3 Port Enable Switches

There shall be one Port Enable switch for each port which will either enable or disable that particular port.

3.8.4 Initialize Enable Switches

There shall be one Initialize Enable switch for each port. In the Enable position the system initialize signal generated by the system controller on that port shall be accepted. However, when the Initialize Enable switch is in the OFF position, the system initialize signal received on that port shall be rejected. Disabling a port shall also reject a system initialize received on that port.

3.8.5 Size Switches

Each port shall have one Size switch which shall indicate that either the total memory assigned to that port is available or that only half the assigned memory is available.

3.8.6 Processor Number Switches

The two processor number switches shall indicate the number assigned to the processor (in a multiprocessor system). They shall have no affect on the processor logic, but shall provide a program number unique to the processor, readable by the RSW instruction.

3.8.7 Fault Base Switches

(To be supplied later)

3.8.8 Initialize and Clear Pushbutton

The Initialize and Clear Pushbutton shall cause all control bistable and data registers in the processor to be reset to a known initialized state. This button shall cause the same action as the initialize signal through a system port.

3.9 OPERATORS PANEL

An operators panel shall be provided on the processor cabinet. This panel design is common to all cabinets and is described in the GE-645F System EPS-1. The controls and indicators are summarized as follows:

- AC Breaker ON indicator
- Power ON pushbutton and indicator
- Power OFF pushbutton and indicator
- Ready/Trouble indicator
- Normal Test indicator
- Over Temperature indicator
- Alarm Reset Pushbutton and indicator
- Audible Alarm

3.9.1 Ready/Trouble Indicator

The Ready/Trouble indicator is defined separately for each system module. For the processor, the trouble condition shall indicate the processor is in either of the following states and hence available to the system:

- Step mode and not in Test (use time meter signal absent)

The Ready condition is the opposite of Trouble.

3.9.2 Audible Alarm

The audible alarm shall be sounded for either of the following conditions:

- Trouble
- Over Temperature

3.10 MAINTAINABILITY AIDS

The maintainability aids provided in the processor shall be of the following general types:

- Maintenance Panel
- History Register
- Program Access and Control of T&D Aids

3.10.1 Maintenance Panel

The maintenance panel for the 645F Processor shall be similar to the 655/EIS Processor maintenance panel. The following additions shall be displayed:

- Register display shall be expanded to include the 645 registers and additions to registers. This includes individual associative memory registers.
- Register initialization shall be expanded where possible.

- Address switches for reading and writing a storage location shall be expanded to 24 bit addressing.

3.10.2 History Register

The 645F shall provide some "recent history" keeping method for the appending unit similar to that provided for the CU, OU and DU as described in the 655/EIS EPS-1.

3.10.3 Program Accessible T&D Aids

(To be supplied later)

3.11 MEASUREMENT AIDS

The following aids to system measurement shall be provided:

- High resolution (1/512 millisecond) Timer Register and (1 microsecond) Elapsed Time Clock (SCU).

The ETC shall be extended from 36 to 52 bits in the 645F and shall have a resolution of 1 microsecond, running in the MULTICS Mode. (The ETC shall be read by the processor but is located in the system controller.)

4. HARDWARE INTERFACES

The 645F Processor, whether operating in MULTICS or GECOS Mode, shall have identical interfaces with: the 645F System Controller, the 635 Master Console, and the measurement aid (if a freestanding monitor is shown to be feasible). The 645F Processor shall not be capable of interfacing with the 655 or 635 System Controller because of the incompatibilities of the Read/Alter/Rewrite cycles.

4.1 SYSTEM PORT MODUIARITY AND INTERFACES

The 645F System Port Interface shall be identical to the 655/EIS System Port Interface with the 655/EIS System Controller (655/EIS EPS-1) except for:

- The addition of a "key" line (see below).
- The addition of four more system ports.

4.1.1 Summary of 645F System Controller Interface Lines

4.1.1.1 Bidirectional Signals

- 72 data lines
- 2 data parity lines

4.1.1.2 Signals to System Controller

- 8 zone lines
- 18 address lines
- 5 command lines

ZAC lines

- ZAC Parity
- $INT access request pulse lines to system controller
- $XMT transmit to port board receive register pulse line
- One key line

4.1.1.3 Signals to Processor

- 3 pulse lines ($PIN, $DA, $CON)
- 1 XIP (interrupt present)line
- 4 illegal action lines
- 1 initialize line
- $XMT line
- 1 ready line

4.2 OPERATIONS WITH THE 645F SYSTEM CONTROLLER

The operation of the 645F Processor and the 645F System Controller in MULTICS Mode shall differ from the operation in GECOS Mode in two areas - addressing and key/lock logic.

4.2.1 Addressing

Operating in GECOS Mode, addressing shall be the same as specified in 655/EIS Processor EPS-1 and the addressing range will include up to 256K (36 bit words) utilizing up to four system ports.

In the MULTICS Mode, addressing shall be extended from 18 to 21 bits (with internal logic expanded to 24 bits) and up to eight system ports. Each of the system ports shall have up to 256K 36 bit words. At some later date, by expanding the port boards to a 21 bit address capability, the 645F Processor operating in MULTICS Mode, shall be capable of addressing up to 2M 36-bit words on each of its eight ports.

The 645F Processor shall be responsible for directing the interrupt to the proper system controller and for detecting an illegal address based on the individual memory size of each controller and on the total memory connected to the processor, respectively.

4.2.2 Key/Lock Logic

A 645F Processor, operating in MULTICS Mode, shall provide a "locking" mechanism for the three cycle Read/Alter/Rewrite command.

During the time period in which a processor (or other active device) is executing a Read/Alter/Rewrite, the "lock" shall inhibit all other devices attempting a Read/Alter/Rewrite until the one in execution is completed.

4.3 INTERFACE ERROR CHECKING

Same as in the 655/EIS Processor EPS-1 (43A232719).

4.4 PROCESSOR INITIALIZATION

The 645F Processor shall function like the 655/EIS Processor when initialized except that the control bistables and data registers in the appending unit shall also be cleared.

4.5 635 MASTER CONSOLE INTERFACE

Same as in the 655/EIS Processor EPS-1 (43A232719).

5. GENERAL DESIGN REQUIREMENTS

The 645F Processor shall conform to the general design requirements contained in the GE-655 General Design Requirements EPS-1 (Applicable Documents, Section 1.2). In particular, like the 655/EIS Processor the logic shall be in a 10 bay cabinet rather than eight bays with the power supply in the lower two bays.

6. RELIABILITY AND MAINTAINABILITY

The 645F Processor shall conform to the EPS 655 Maintainability Design (Reference Documents, Section 1.2).

6.1 RELIABILITY OBJECTIVES

6.1.1 Availability

The availability objective shall be a minimum of 99 percent where:

$$\text{AVAILABILITY} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \times 100$$

or $\text{MTBF} \geq 99\ \text{MTTR}$

See GE-655 Maintainability Design EPS, 43A219617 for detailed availability information.

6.1.2 Mean Time Between Failures

The minimum acceptable MTBF shall be in accordance with the standards set by the GE-655 Maintainability Design EPS, 43A219617.

6.2 TRANSIENT ERROR RATE

Same as in the 655 Processor EPS-1.

6.3 MEASUREMENT OF TRANSIENT ERROR RATE

Same as in the 655 Processor EPS-1.

6.4 MAINTAINABILITY REQUIREMENTS

Comprehensiveness - Minimum of 95 percent on new circuit boards.

Diagnosability - Minimum of 98 percent on each board.

Resolution - Average 2.5 chips 95 percent of the failures must have a resolution of 10 or less chips per board.

INSTRUCTION INDEX

	Mnemonic	Name of Instruction	Op Code(octal)	Page
A	ABSA	Absolute Address to Accumulator	212 (0)	106
	ADWPn	Add to Word Register of PRn	050 (0)	117
			051 (0)	
			052 (0)	
			053 (0)	
			150 (0)	
			151 (0)	
			152 (0)	
			153 (0)	
C	CALL	Call	713 (0)	127
	CAMP	Clear Associative Memory-Paged	532 (1)	129
	CAMS	Clear Associative Memory-Segmented	532 (0)	128
	CIOC	Connect I/O Channel	015 (0)	134
E	EASPn	Effective Address to Segment Number Field of PRn	311 (0)	101
			310 (1)	
			313 (0)	
			312 (1)	
			331 (0)	
			330 (1)	
			333 (0)	
			332 (1)	
	EAWPn	Effective Address to Word/Bit Number Field of PRn	310 (0)	102
			311 (1)	
			312 (0)	
			313 (1)	
			330 (0)	
			331 (1)	
			332 (0)	
			333 (1)	
	EPAQ	Effective Pointer to AQ Register	213 (0)	105
	EPBPn	Effective Pointer at Base to PRn	350 (1)	104
			351 (0)	
			352 (1)	
			353 (0)	
			370 (1)	
			371 (0)	
			372 (1)	
			373 (0)	
	EPPn	Effective Pointer to PRn	350 (0)	103
			351 (1)	
			352 (0)	
			353 (1)	
			370 (0)	
			371 (1)	
			372 (0)	
			373 (1)	

	Mnemonic	Name of Instruction	Op Code(octal)	Page
L	LDBR	Load Descriptor Segment Base Register	232 (0)	95
	LDI	Load Indicator Register	634 (0)	138
	LPRI	Load Pointer Registers from ITS Pairs	173 (0)	94
	LPRPn	Load Pointer Register n Packed	76n(0)	93
	LPTP	Load Page Table Pointers	257 (1)	99
	LPTR	Load Page Table Registers	173 (1)	98
	LRA	Load Ring Alarm	774 (1)	96
	LSDP	Load Segment Descriptor Pointers	257 (0)	100
	LSDR	Load Segment Descriptor Registers	232 (1)	97
M	MME2	Master Mode Entry 2	004 (0)	131
	MME3	Master Mode Entry 3	005 (0)	132
	MME4	Master Mode Entry 4	007 (0)	133
R	RCCL	Read Calendar Clock	633 (0)	130
	RCU	Restore Control Unit	613 (0)	125
	RET	Return	630 (0)	135
	RMCM	Read Memory Controller Mask Register	233 (0)	130.1
	RSW	Read Switches	231 (0)	130.4,5,6
	RTCD	Return Double	610 (0)	126
S	SCU	Store Control Unit	657 (0)	121
	SDBR	Store Descriptor Segment Base Register	154 (0)	111
	SMCM	Set Memory Controller Mask Register	553 (0)	130.3
	SMIC	Set Interrupt Cells	451 (0)	130.2
	SPBPn	Store Segment Base Pointer of PRn	250 (1) 251 (0) 252 (1) 253 (0) 650 (1) 651 (0) 652 (1) 653 (0)	110
	SPRI	Store Pointer Registers as ITS Pairs	254 (0)	108
	SPRIn	Store Pointer Register n as ITS Pair	250 (0) 251 (1) 252 (0) 253 (1) 650 (0) 651 (1) 652 (0) 653 (1)	109

Mnemonic	Name of Instruction	Op Code(octal)	Page
SPRPn	Store Pointer Register n Packed	54n (0)	107
SPTP	Store Page Table Pointers	557 (1)	115
SPTR	Store Page Table Registers	154 (1)	113
SRA	Store Ring Alarm	754 (1)	116
SREG	Store Registers	753 (0)	137
SSDP	Store Segment Descriptor Pointers	557 (0)	114
SSDR	Store Segment Descriptor Registers	254 (1)	112
STAC	Store a Conditional	354 (0)	118
STACQ	Store a Conditional-C(Y) = C(Q)	654 (0)	119
STC1	Store Instruction Counter plus 1	554 (0)	136
STCD	Store Control Double	357 (0)	120
STI	Store Indicator Register	754 (0)	139
TEO	Transfer on Exponent Overflow	615 (0)	147
TEU	Transfer on Exponent Underflow	615 (0)	146
TMI	Transfer on Minus	604 (0)	148
TNC	Transfer on No Carry	602 (0)	144
TNZ	Transfer On Not Zero	601 (0)	151
TOV	Transfer on Overflow	617 (0)	145
TPL	Transfer on Plus	605 (0)	142
TPNZ	Transfer on Plus and Nonzero	602 (1)	150
TRA	Transfer Unconditionally	710 (0)	140
TRC	Transfer on Carry	603 (0)	143
TRTF	Transfer on Truncation Indicator OFF	601 (1)	155
TRTN	Transfer on Truncation Indicator ON	600 (1)	149
TSPn	Transfer and Set PRn	270 (0) 271 (0) 272 (0) 273 (0) 670 (0) 671 (0) 672 (0) 673 (0)	124
TSS	Transfer and Set Slave	715 (0)	156
TSXn	Transfer and Set Index Register n (n = 0, 1, ..., 7)	70n (0)	141
TTF	Transfer on Tally Runout Indicator OFF	607 (0)	153
	Transfer on Tally Runout Indicator ON	603 (1)	154
TZE	Transfer on Zero	600 (0)	152

Honeywell Rev.

Cont. on Page Final Page 185

HONEYWELL INFORMATION SYSTEMS INC.
Phoenix Computer Operations

TITLE: 645 FOLLOW-ON CPU, ENGINEERING PERFORMANCE SPECIFICATION (EPS-1)

Prepared by: R. H. Dobberstein — Sept 24, 1971
R. H. Dobberstein, Date
6000 CPU Design Engineering

Approved by: C. W. Dix — 10-12-71
C. W. Dix, Date
Director of Engineering

Reviewed by: R. L. Ruth — 9/24/71
R. L. Ruth, Manager Date
Computer Design Engineering

Reviewed by: J. M. Risan — 09-24-71
J. M. Risan, Manager Date
6000 CPU Design Engineering

Reviewed by: C. T. Clingen — 9/29/71
C. T. Clingen, Manager Date
Cambridge Information Systems Lab.